在 Hadoop 中以 Failover Controller 改進 AvatarNode 可用性的架構

李滄智 [1]　　高勝助 [2]
[1] 國立中興大學 資訊科學與工程學系
w99056019@cs.nchu.edu.tw
[2] 國立中興大學 資訊科學與工程學系
sjkao@cs.nchu.edu.tw

摘要

Hadoop 採用 HDFS 系統，以一台 NameNode 與多台 DataNodes 提供計算與儲存的叢集環境。NameNode 類似於 Master 的身份，負責處理客戶端的存取請求與儲存整個 HDFS 中各檔案的 metadata ；而 DataNode 負責實際資料的儲存。當 NameNode 發生故障時，Hadoop 將無法提供服務，而此時暫存在 NameNode 記憶體中的資料儲存位置資訊，會因此遺失，此即為單節點失效(Single Point of Failure)的問題。本文中，我們延伸 AvatarNode 為解決 NameNode 單節點失效問題的架構，在 Primary 與 Secondary NameNode 之外，增加了一個 Spare NameNode 的設計，實作出一個 NameNode 高可用性的叢集架構。我們以 Failover Controller 作為架構的核心，監控各節點的狀態。當 Active NameNode 出現異常時，自動切換 Standby NameNode 為 Active，並將 SpareNode 轉換為 Standby。基於我們設計的改良架構，從實驗結果得知可達到自動切換 NameNode，並在 3 分鐘內完成接手，並且不會在切換過程中導致資料遺失，此為本文增強 NameNodes 可用性的特點。

關鍵詞：雲端運算、Hadoop、AvatarNode、Failover。

1. 前言

Hadoop Distributed File System (HDFS) [1]就是由 Google 雲端架構[16]得到啟發而開始的開放原始碼計劃，Hadoop 是 Apache 專案中之一的項目，包含了 HDFS、MapReduce[7]，是一種採用 Java 平台打造的分散式資料處理技術，用來處理與保存大量資料在雲端，在最近幾年來，因為其能夠處理網際網路與其他產業面臨的巨量資料問題，因此在網路世界上頗受注目。而在社群網站的崛起後，大量資料的存取運算更顯得重要，如何蒐集並運用成為現今網路平台的一大課題。目前許多大型網站，如 Amazon、Facebook、Yahoo[8]都已經應用了 Hadoop 在其服務中，它的設計規模從一台服務主機到成千上萬的機器，每個地方提供計算和儲存。在 2006 年 Hadoop 問世後，短短幾年時間成長迅速，然而從 Hadoop 官方提出的架構得知，叢集裡包含著多個 Slave (DataNode & TaskTracker)，而管理這些節點的 Master(NameNode & JobTracker) 被設計成單節點；亦即當 Master 節點失效時，會導致整個 HDFS 架構失效；此種狀況，我們稱作「Single Point of Failure」，以下我們簡稱 SPOF[2]。因此為了達到 High Availability Cluster[15]，完善的應變方法是必須的。

然而 Hadoop 發表起初本身並沒有針對單節點失效 SPOF 的問題有直接的解決方法；但許多大型網站如社群網路 Facebook 也應用 Hadoop 於後端的資料關聯分析，因此為了維持他們系統營運服務不中斷的問題，他們於 2010 年提出 AvatarNode[3]以解決此項問題，其重點在於不貿然異動整個系統，而可以有效解決，然仍有部分功能未臻完善，例如沒有實作自動切換 Standby NameNode 至 Active，仍需人工介入，手動切換 NameNode，也就是所謂 Cold Backup[4]的作法。本論文研究從 AvatarNode 架構著手，繼承其架構並實作監控機制，以盡量保留沿用系統功能為考量，加入 Failover Controller(以下簡稱 FC)進行監控，NameNode 異常時，能有效率地進行失效 NameNode 節點的切換，並快速地重新啟用服務，使原先正在使用服務的 client 端，能自動重新對應新的 NameNode，維持服務的高可用性，以達到所謂的 Hot Backup[4][6][13]的功能。

本文結構如下。首先在說明 Hadoop 的背景知識，以了解其架構原理，並描述各子項目內容。後續會提出我們採用的實作架構，針對各流程加以解析並說明。最後章節談論未來展望。

2. 相關技術背景

2.1 Hadoop Distributed File System

Hadoop Distributed File System 的設計理念是在分散式的儲存環境裡，是一個採用串流資料存取模式的檔案系統。採用了主從式(Master/Slave)的架構，由一個 NameNode 和多個 DataNode 所組成，並且由於是採用 Java 語言所開發，因此任何支援 Java 平台的機器都可以佈署。

HDFS 擁有許多特性，如支援「超大型檔案」，

可以處理檔案大小多達 GB、TB 抑或是 PB 等級的資料。且不需要昂貴的硬體，一般電腦即可佈署，在大型叢集環境中，若是某節點故障，仍可以正常運作；亦可以處理「大量的小檔案」，並且其分散式檔案系統具備的資料存取特性為 Write Once Read Many 存取模式。

具有資料的一致性。Client 端在檔案沒有寫入完成之前，是無法看到檔案地存在的。檔案會根據設定，被切割為多個區塊儲存。並被分配到適當的節點上存放。預設的磁碟區塊(Block)大小為 64MB。如此倘若某區塊損壞，仍可由其他副本重新取得，保持資料的完整性。

2.2 NameNode 與 DataNode

一個 HDFS 叢集是有一個 NameNode 和多個 DataNode 組成。這兩種節點實作 Master 與 Worker 的工作。NameNode 管理檔案系統的 namespace，維護檔案系統的樹狀結構和在樹中的所有檔案和目錄的 metadata，並執行 namespace 的操作，例如 open、close 與 rename 等等。

而這些資訊會以兩種格式存在於 NameNode 中，一個儲存在本機的磁碟空間上為 namespace image (FSImage)。另一個為存放在記憶體空間的 edit log(edits)[17][18]。

DataNode 則是負責儲存資料的地方。會透過 Heartbeat 定時向 NameNode 發送所儲存的檔案區塊資訊。

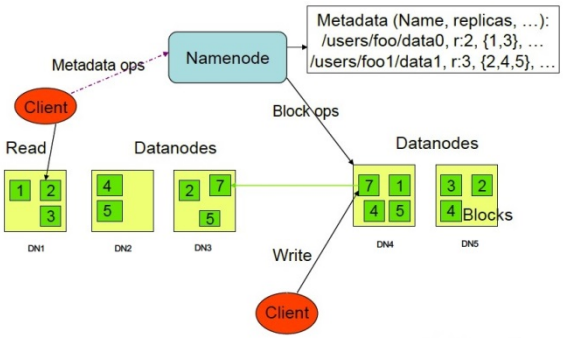


圖 1 HDFS 架構圖[9]

如圖 1 架構圖所示，NameNode 本身記錄著兩項重要的資訊，Metadata 與 DataNodes 的 Block 位置訊息。Metadata 包含兩項資訊：資料名稱與副本數量；Block 位置訊息則記錄著各 DataNodes 所存放的各資料內容，該資料儲存在 NameNode 的記憶體，如此才能快速搜索到資料所存放的位置。然卻因為如此，當 NameNode 服務重新啟動時，會花費相當多的時間在重新接收所有 DataNodes 所傳送過來的 Block 位置訊息。當 DataNodes 越多等待時間越長，這是 NameNode 啟動時耗時最久的動作。

2.3 AvatarNode

Hadoop AvatarNode High Availability[10]，Facebook Hadoop 架構師(Dhruba Borthakur)於 2010 年提出，當時針對 SPOF 並沒有太多做法。但經過他們團隊的分析現狀和可能的解決方案，設計了一個簡單的解決方案，可以達到最小的變動且可達到當時需求。

AvatarNode 完全封裝了 NameNode，並實作自己的介面，以提供所需要的改良。AvatarNode 執行有兩種模式 Active Avatar 或者 Standby Avatar。如果啟動執行在 Active Avatar 模式，那麼它就和當前 NameNode 功能完全一樣，延伸原有的 Hadoop NameNode 的原有功能。其實執行的動作就是 NameNode 所提供的 API。主要架構如圖 2 所示，執行 Active Avatar 模式的 AvatarNode 機器，儲存 FSImage 到一個共享的 NFS。在另外一台機器上，啟動 AvatarNode 的另一個實例，啟動時執行在 Standby Avatar 模式。

Standby AvatarNode 封裝了 NameNode 和 Secondary NameNode，並提供了兩個類別所擁有的介面。從圖右上角區塊得知，Standby AvatarNode 持續的從共享的 NFS filer 中讀取 HDFS editlog，並持續的把這些 log 推送到 Standby AvatarNode 中 NameNode 實例。Standby AvatarNode 中的 NameNode 執行在 SafeMode。SafeMode 是原 NameNode 本身的一種狀態，正常 NameNode 啟動時，會先進入 SafeMode，等接收所有 DataNode 的 BlockReport 資訊後，才會脫離 SafeMode。而因為 Standby AvatarNode 不可以同時對 NFS filer 裡的檔案做異動，會導致兩個 NameNode 同時對同一個檔案做異動，因此不能讓 Standby AvatarNode 負責 Primary 的工作，但卻必須保持和 NameNode 同步的 Metadata 資訊。而這一個部分，就是改善 Standby AvatarNode 切換為 Active 時，大幅改善啟動速度的最大關鍵。

HDFS 客戶端訪問 NameNode 通過一個虛擬 IP(Virtual IP Address)。當發生故障需要快速切換時，管理員會需要手動在機器上刪掉 Active AvatarNode process，然後在另一個機器上執行將 Standby AvatarNode 切換到 Active 狀態，使其離開 SafeMode，並接管原先應該執行的相關任務。新的 Active NameNode 會保證把所有 request 的 job 都處理完，因為此時會重新打開 edit log，並處理完文件中所有的任務。這個假設是基於 NFS-v3[12] 支持 close-to-open cache coherency semantics。Standby AvatarNode 把所有 NFS filer 上的事務處理完之後，退出 SafeMode 模式。管理員將第二台機器的 VIP 換為原先機器的 IP，這樣所有的 HDFS 客戶端的請求會送到新的機器上。

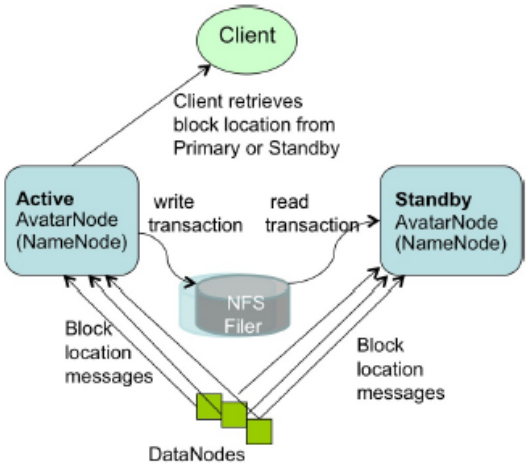


圖 2 AvatarNode 架構圖[11]

但此做法最大問題仍是需要人工介入，需要透過管理者去執行切換的動作已達到 failover。因此下一章節實作部分會針對這問題加以解決。以提高 Hadoop 的高可用性。

3. 高可用性的 Avatar 架構

本章節將分析 AvatarNode 實作的架構並加入本論文提出的修改方法，透過建立三台 NameNode 的小型叢集，達成 Hadoop 的高可用性。其中包括 Active NameNode、Standby NameNode、Spare NameNode，下面的章節會逐步說明架構細節。從類別架構到流程說明逐一闡述本架構運作方式。(下文中 Standby NameNode 簡寫為 StandbyNode，Spare NameNode 簡寫為 SpareNode)。

3.1 Avatar 改進策略

AvatarNode 的優勢在於能夠快速切換 NameNode。但其缺點在於需要手工切換 StandbyNode；NameNode 異常時，記錄記憶體中尚未寫入 FSImage 的 blockreport，無法被新的 Active NameNode 所讀取，會因此導致部分資料的遺失；因此我們針對這些問題進行改良設計：

(1) 自動切換

原 AvatarNode 沒有能夠偵測 NameNode 狀態的方式，因此在 NameNode 中常駐 FCAgent(下文會提到實作方式)，用以監控 NameNode 彼此存活狀態，並記錄各個 NameNode IP 位址，讓 client 端傳送請求時，能夠得到完整的 NameNodes IP。透過 FCAgent 在 NameNode 異常發生時，進行所謂的 NameNode 自動切換功能。

(2) 資料完整性

在 NameNode 發生異常時，原記錄在記憶體中的資料，editlog 尚未寫入 NFS，也就代表說，當下的 StandbyNode 並無法取得該資料，因此在接管 NameNode 後，此部份的 editlog 會遺失。我們的作法讓 Active NameNode 與 StandbyNode 都確保 editlog 已經記錄在各自的記憶體中，才算完成一段完整的動作。讓異常發生時，NameNodes 記憶體內的 editlog 保持一致。

(3) NameNodes 穩定性

AvatatNode 手動切換後，原先的 NameNode 無法重新再啟動為 StandbyNode。因此我們透過 FCAgent 可以設定重新啟動的 NameNode 所扮演的角色，並加入另一個我們稱為 SpareNode 的 NameNode，用以監控 StandbyNode，以為維持 NameNodes 的穩定性。

3.2 系統架構

我們沿用原 AvatarNode 架構，延伸一台 SpareNode，如圖 3 中的架構所示。平時 SpareNode 處於 SafeMode 狀態，隨時監控 StandbyNode，於 NameNode 失效時，StandbyNode 將會轉換成 Active 的 NameNode，並喚醒 SpareNode 以扮演 StandbyNode 的角色；在原先失效的 NameNode 修復重新啟動後，因為發現已有 Active 的 NameNode 和 StandbyNode 存在，因此，將會成為在 SafeMode 狀態的 SpareNode。

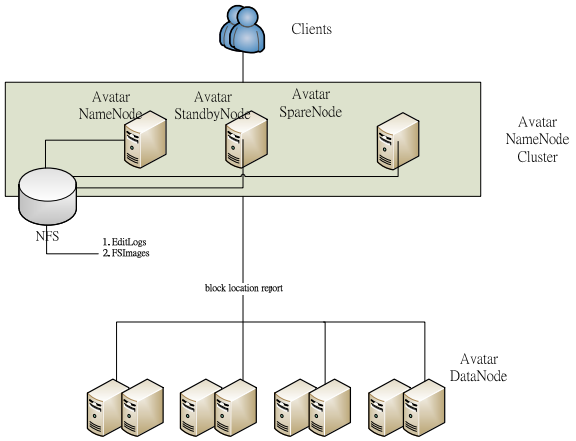


圖 3 實體架構圖

本架構主要以納入 SpareNode 以強化 AvatarNode 在 NameNode 失效時的高可用性，並提高架構的穩定性。系統主要的運作功能如下:

1 各 NameNode 監控彼此狀態。

2 StandbyNode 監控 Active NameNode，若偵測 NameNode 異常，自動切換接手 Active NameNode 的角色。

3 多一個 Node 監控 StandbyNode 目前狀態，若發生異動，以接手 StandbyNode 所扮演的的角色。

4 穩定維持 NameNode cluster 中三台

NameNodes 運作的狀態。

3.3 Failover Controller 設計方法

欲達成自動管控 NameNode，並維持原先架構，我們以最小幅度設計規劃以下模組，以達成 failover 機制。首先為了監控每個 node 狀態，我們先設計建立 ResourceManagement 介面與 FailoverController 等介面，如下圖 4，並撰寫實作的類別 FCAgent。

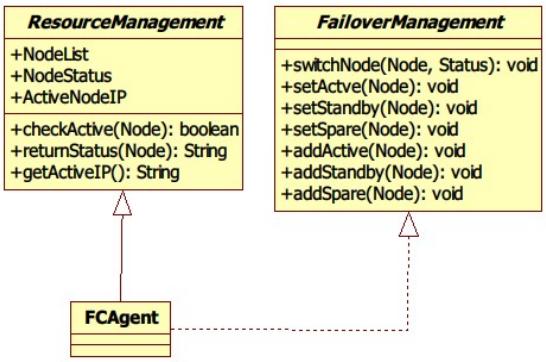
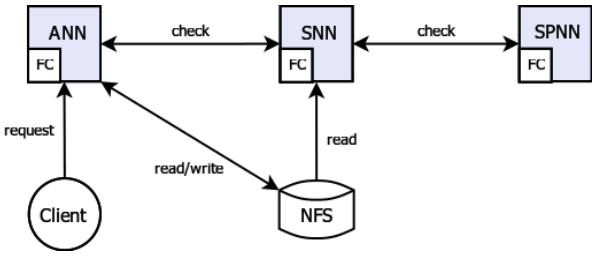


圖 4 Failover Controller 類別圖

由 ResourceManagement 介面負責處理監控 Node 彼此狀態，記錄目前的 Active NameNode IP 位址，讓要求服務的 client 端，在每次傳送要求時，都會得到 Active NameNode 的 IP 位址，隨時記錄目前可以取得服務的 NameNode。三個 NameNode 並透過 checkActive 類似 heartbeat 的方式確認彼此存活狀態，達成監控的機制。透過取得的狀態決定是否做切換的動作。當狀態發生異動時，透過實作 FailoverController 介面，透過 switchNode 方法處理切換 NameNode 的行為，並註冊目前 Active NameNode IP 位址，使用 setActive 將 StandbyNode 註冊為 Active NameNode。而 client 端在處理傳送要求時，可從 Active NameNode 取得目前 Active 與 Standby NameNode 的 IP 位址，當 client 端發現無法從 Active NameNode 取得 response 時，會改向 StandbyNode 發出請求，直到 NameNode 完成 switchNode 切換動作，並重新註冊新的 Active NameNode，client 端發出的要求被新的 Active NameNode 接受後，得到的 response 中，會記錄著 Active NameNode 已經切換。

而接下來 SpareNode 檢查到 StandbyNode 的異動後，也透過 FCAgent 觸發 switchNode 的動作，將本身的狀態切換成 StandbyNode 並註冊，再透過一次 checkpoint[5]將所有 block reports 存進記憶體，如此即可成為 StandbyNode。此時 cluster 仍舊維持一個 Active NameNode 與 StandbyNode 存活的狀態。

原先異常的 Node 若經過恢復後，可將其設為 SpareNode 啟動，此時便回復一開始的狀態。如下圖 5 的架構圖，即為本文針對 SPOF 主要解決架構。FCAgent 為實作圖 4 ResourceManagement 與 FailoverController 的介面，並常駐在各 node 中。達到監控並切換的機制，此為本文 Failover Controller 機制的處理架構。



ANN : Active Name Node	SNN : Standby NameNode
SPNN : Spare NameNode	FC : FCAgent

圖 5 Failover 策略

在正常情況下，如圖 5，ANN 與 SNN 是通過共享儲存 editlog，儲存裝置以 NFS 方式掛載，讓兩個 NameNode 同時可以存取同一份 FSImage。而原先 AvatarNode 策略已經確保兩個 NameNode 都能記錄 DataNode 所回報的 blockreport，皆存放在各自的記憶體中，透過這樣的先決條件，在 SNN 實際轉換成 ANN，可避免等待所有 DataNode 作 blockreport 的時間。

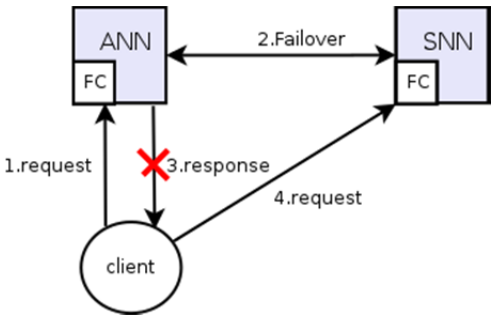


圖 6 Failover on client side

另外本架構在 client 端會同時紀錄兩個 NameNode 的存在，並擁有 Retry 機制，當 client 端收到 NameNode 丟出的異常、或是無法偵測到 NameNode 的存活狀態時，代表當下 NameNode 可能已經中斷服務，如圖 6，而 StandbyNode 正在進行切換中，在此種狀態下是無 NameNode 的狀態；為保持 client 所傳送的要求不會被中斷或遺失，client 端會轉向原紀錄的 StandbyNode 發送請求，直到 StandbyNode 接管成為 Active NameNode 為止，此方式是為了讓 client 端在 Failover 過程中，不會因為無 NameNode 的狀態，導致 client 端的請求失敗或遺失。

此架構主要增進部分做法以完善 AvatarNode 在自動切換 NameNode 的不足，並提高其架構穩定性。並以最小幅度的系統異動，避免已經運作的系統大幅度的調整，即可解決 SPOF 問題，增加 NameNode 的可用性。

3.4 測試與分析

測試環境包含五台 Inter Xeon CPU E5420 2.50GHz VM，一個 Active NameNode、StandbyNode 、SpareNode 與兩台 DataNode。還有兩台 client 端。

本實驗主要測試 NameNode 可用性。透過 checkActive API 去檢查 Active NameNode 與 StandbyNode 狀態，從得到的回應資訊可以取得目前 Active Node 是否正常運作中，實驗步驟如下：

(1) 分別啟動 Active NameNode 與 StandbyNode。
(2) 透過測試程式每秒呼叫一次 checkActive，兩台 Node 同時監控。
(3) 停掉 Active NameNode 的程序。
(4) StandbyNode 自動切換程序啟動。
(5) 驗證 checkActive 取得的 Active Node 是否正確。

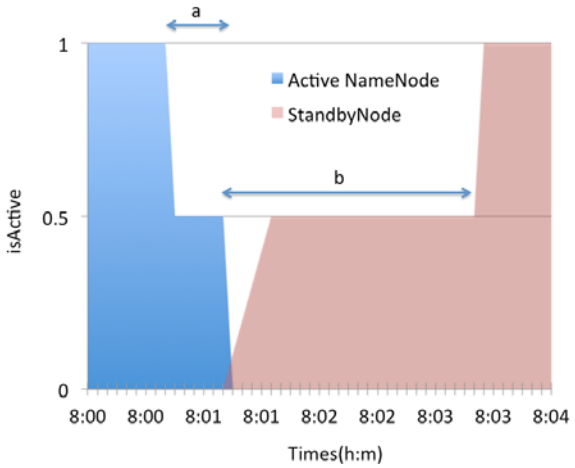


圖 7 NameNode 切換時間

實驗數據如圖 7，縱軸格式數據 1 代表 Active 啟用中， 0 代表目前不是 Active NameNode。橫軸為時間格式，每秒記錄一次。在 a 區間為 checkActive 開始偵測不到 Active NameNode 正確的回應，預設為 30s 後，FCAgent 便會動作，執行 switchNode，自動切換 StandbyNode 為 Active 狀態。而 b 區間即為 StandbyNode 啟動接手所耗費的時間。本實驗在 3 分鐘以內即完成自動接手動作，主要依靠 AvatarNode 共用 NFS 的架構與本文加入的 FCAgent 監控，方能達成快速自動接管，以增加 NameNode 的高可用性。

另一個實驗主要在測試資料完整性，在 client 端傳送 50 個 100MB 的寫入 HDFS，針對不同版本的 Hadoop 重複測試，並在寫入過程中，停掉目前 Active NameNode 的服務，由 StandbyNode 接手，表 1 為測試結果。

因為在切換 NameNode 時，會有一段時間沒有 NameNode 的服務，因此有可能導致 client 傳送資料的遺失。而表 1 中原生 Hadoop0.20.2 在原 NameNode 存在記憶體中的 editlog 會因為尚未執行 checkpoint 的備份而導致遺失，因此切換前的資料會因此而遺失。而 Avatar 的策略仍需要手動切換 NameNode，雖然啟動接管速度較快，但仍有可能有部分遺失，如表所示有一份文件的遺失。

因此本架構在異常發生時可以實現自動快速切換，無須人工干預，避免切換過程的不穩定，如此可在切換完成後及時接收 client 的請求，這些都是改進後的優勢。

表 1 寫入資料測試

Proposal	*文件數量*	*切換後儲存文件數量*	*遺失文件*
Hadoop0.20.2	50	45	5
Avatar 策略	50	49	1
Avatar with FC	50	50	0

4. 結論與未來發展

Hadoop 分散式運算架構，已經應用在很多方面，Facebook 亦應用在廣告效益分析等應用，而實際運作 Hadoop 分散式集群各家有不同的做法，AvatarNode 亦是 Facebook 提出的一種方式，然仍有部分尚未完成的階段；本論文的設計修正了 AvatarNode 需要人工啟動 StandbyNode 的問題，新增 FCAgent 控管資源，透過註冊表的方式記錄在此 NameNode cluster 中所有 Node 的狀態，保持擁有 StandbyNode 來支持熱備援的機制；針對 NameNode 所記錄的 metadata 改存放置 NFS，以方便與 StandbyNode 共享 metadata，而 Avatar DataNode 除了將區塊紀錄位置回報到 NameNode 外，亦會回報給 StandbyNode，以存放在記憶體裡，達到切換到 Active 時可以快速接手啟用服務，有效並快速解決單節點失效自動備援的問題。從 Hadoop 官方的規格透露出，官方已有自己規劃 Hadoop NameNode HA cluster，但在未發表之前，本論文的作法仍有許多發展空間。

本論文中仍有許多可以再修改的部分，例如 NameNode cluster 的穩定性、cluster 的設定。下一個階段亦可拿掉 SpareNode，改由許多 StandbyNode 來做熱備援，可以不限制 StandbyNode 的數量，更可以達到集群設定的彈性與 NameNode cluster 的自由度，亦可自由擴充 NameNode cluster 以符合實際需求。而官方提出的 Warm HA NameNode going Hot 規格[14]，提出的 LoadReplicator 作法，亦是將來可以參考的設計方向。

參考文獻

[1] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The Hadoop Distributed File System,” in Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium, May 2010, pp. 1 –10.

[2] Hadoop Wiki, “NameNode Failover, ” http://wiki.apache.org/hadoop/NameNodeFailover , last access in May, 2013.

[3] D. Borthakur, “Hadoop AvatarNode High Availability,” http://hadoopblog.blogspot.tw/2010/02/hadoop-namenode-high-availability.html , last access in January, 2013.

[4] Andre Oriani,Islene C. Garcia,Rodrigo Schmidt,"The Search for a Highly-Available Hadoop Distributed Filesystem," Facebook, Inc., Palo Alto, CA, USA, August 2010

[5] Dilbag Singh, Jaswinder Singh, Amit Chhabra, "High Availability of Clouds: Failover Strategies for Cloud Computing Using Integrated Checkpointing Algorithms," Communication Systems and Network Technologies (CSNT), 2012 International Conference, May 2012, pp 698-703

[6] Andre Oriani, Islene C. Garcia, "From Backup to Hot Standby: High Availability for HDFS," Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium, October 2012, pp 131-140.

[7] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” in Operating System Design and Implementation, 2004, pp. 137–150.

[8] “Yahoo! Launches World’s Largest Hadoop Production Application,“ http://developer.yahoo.com/blogs/hadoop/posts/2008/02/yahoo-worlds-largest-production-hadoop/ , last access in May, 2013.

[9] “HDFS Architecture Guide ,” http://hadoop.apache.org/common/docs/stable/hdfs_design.html , last access in May, 2013.

[10] “Hadoop AvatarNode High Availability, “ http://hadoopblog.blogspot.com/2010/02/hadoop-NameNode-high-availability.html , 2010, last access in May, 2013.

[11] Dhruba Borthakur. Kannan Muthukkaruppan. Karthik Ranganathan. Samuel Rash. Joydeep Sen Sarma, “Apache hadoop goes realtime at Facebook,” Association for Computing Machinery New York, USA , 2011.

[12] “NFS-V3,“ http://en.wikipedia.org/wiki/Network_File_System , last access in May, 2013.

[13] “Hot Standby for NameNode,” http://issues.apache.org/jira/browse/HDFS-976 , last access in May, 2013.

[14] “Warm HA NameNode going Hot,” https://issues.apache.org/jira/browse/HDFS-2064 , last access in May, 2013.

[15] “High_availability_cluster,” http://en.wikipedia.org/wiki/High-availability_cluster , last access in May, 2013.

[16] S. Ghemawat, H. Gobioff, S. Leung. “The Google file system,” In Proc. of ACM Symposium on Operating Systems Principles, Lake George, NY, Oct 2003, pp 29–43.

[17] Weijia Xu, Wei Luo, Nicholas Woodward, “Analysis and Optimization of Data Import with Hadoop,” Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International, May 2012, pp 1058-1066.

[18] Feng Wang, Jie Qiu, Jie Yang, Bo Dong, Xinhui Li, Ying Li , "Hadoop high availability through metadata replication," Proceeding CloudDB '09 Proceedings of the first international workshop on Cloud data management, 2009, pp 37-44.