具資料高度保護能力的分散式區塊儲存系統-DISCO

DISCO: Distributed Block-Level Storage System with Comprehensive Data Protection

朱怡虹
Yi-Hong Chu

中文摘要

近年來隨著網路普及和使用者終端設備數量的增加，每天都有大量的資料不斷地產生。快速成長的巨量資料需要一個具備儲存空間擴充能力、資料高度保護能力、和高儲存效率的儲存系統。這幾年有很多分散式儲存系統因此而誕生，其分散式架構使得儲存空間的擴充變的容易，因此可解決每天不斷增加的資料的儲存問題。然而，這些系統大多只提供基本的資料存儲，使用者需自行將這些系統與提供備份功能的次要儲存系統（Secondary Storage System）整合才可以讓資料有完整的保護，但這些整合過程通常耗時且繁雜。因此我們設計了一個新的儲存系統DISCO（Distributed Integrated Storage with Comprehensive Data Protection），此系統提供透過虛擬硬碟存取的區塊式資料儲存，並提供了完整資料保護功能以防範各種可能造成資料毀損的情況，包括:資料N份複本技術（N-way Replication）可避免單一儲存節點損壞造成資料無法存取、虛擬硬碟快照功能可解決人為失誤造成的資料毀損狀況、異地備份功能可解決天然災害導致資料全毀的問題。另外，此系統更透過了自動精簡配置（Thin Provisioning）、資料去複本（De-duplication）、資料和元資料(Metadata)零複製的虛擬硬碟複製技術的實作提高了資料儲存效率。DISCO系統現今已完成上述所有功能開發與測試，且與OpenStack Mitaka版本相容的Cinder驅動程式也已獲得官方認證通過，因此更可作為OpenStack系統的儲存方案。

Abstract

With the popularity of the internet and the increase of intelligent user devices, huge data set is generated every day. The fast-growing big data demands a storage system with high scalability of storage space, high data protection capability, and high storage efficiency. Therefore, in these years tremendous distributed storage systems have been developed to achieve high storage space scalability. However, most of them only provide basic data access. Users need to manually integrate these systems with other secondary storage systems with backup functionalities to get higher data protection; however, the integration is usually time-consuming and complicated. In this paper, we have developed a novel storage system called DISCO (Distributed Integrated Storage with Comprehensive Data Protection) to provide high data protection. DISCO not only provides block-level storage service through virtual disk access, but also provide full data protection to handle most of failure cases. For example, N-way replication technology is adopted to prevent data loss from single storage server failure, the snapshot feature can protect data from unexpected user errors, and the remote backup feature will be helpful for disaster recovery. In addition, DISCO also achieves high storage efficiency by the implementation of thin provisioning, data de-duplication, and virtual disk clone with zero data copy and zero metadata copy. DISCO has released the first stable version, and its Cinder driver for OpenStack Mitaka has been verified by the organization such that DISCO can be taken as the storage solution in OpenStack deployments.

關鍵詞 (Key Words)

區塊式儲存系統（Block-level Storage System）

分散式儲存系統（Distributed Storage System）

儲存虛擬化（Storage Virtualization）

1 · DISCO系統介紹

隨著物聯網科技興起、網路和個人終端設備的普及，成長快速的巨量資料急需一個具高擴充性、資料高度保護能力、和高儲存效率的儲存系統。根據統計[1]，物聯網科技產生的大量數據，估計2018年將會達到高達400ZB規模的數量。還有因為網路普及和上網終端設備增加，雲端儲存空間需求也將日漸增加，CISCO認為至2018年為止將有53%家庭消費者會使用雲端儲存，每人每月平均雲端儲存量也將由2013年的186MB增加至2018年的811MB [2]。

快速成長的巨量資料催生了高擴充性的分散式儲存系統開發，例如: Google File System（GFS）[3, 4]、HDFS[5]、GlusterFS[6, 7]、Lustre[8, 9]提供了使用者以檔案為存儲單位的分散式檔案系統，Ceph[10, 11]、Sheepdog[12]則提供以虛擬硬碟（Volume）為存取媒介的區塊式儲存系統。其中，又以區塊式儲存較為彈性，因為區塊儲存裝置（Block Device）為儲存架構的最底層，使用者可在此區塊儲存裝置上再根據需求格式化成各式檔案系統。然而，現有的區塊式儲存系統大多只提供基本的資料存儲功能，在欠缺大型雲端儲存系統該有的資料備份功能的情況下，資料將無法受到完整保護，例如: Ceph[10, 11]沒有提供自動化異地備份功能，因此當天災發生時，整套系統可能全毀因此資料將全部無法存取。因此我們開發了DISCO（Distributed Integrated Storage with Comprehensive Data Protection），目的在建立一個擁有高擴充性、資料完整保護、和高儲存效率的區塊式儲存系統。

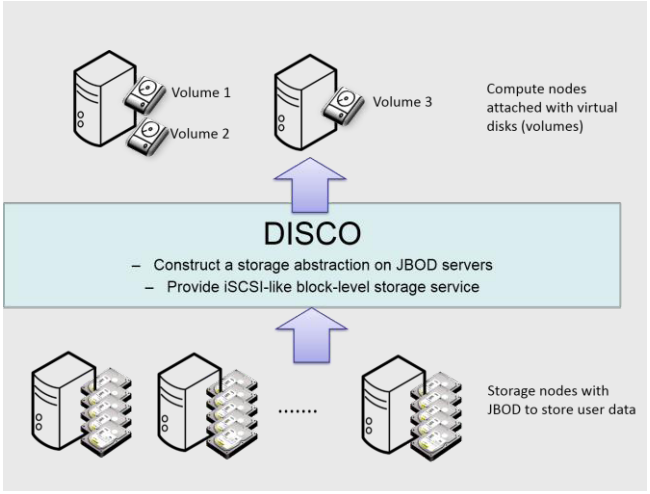

圖 1　DISCO提供的區塊式儲存服務

DISCO是一個分散式區塊儲存系統，如圖1所示，此系統提供使用者區塊式儲存服務，使用者可透過此系統產生虛擬硬碟並將它掛載到有安裝DISCO軟體的運算伺服器（Compute Node）上，接著，使用者就可存取此虛擬硬碟就如同存取一般實體硬碟一樣。此系統使用多個掛滿硬碟（Just a Bunch of Disks；JBOD）的儲存伺服器（Storage Node）來儲存虛擬硬碟資料，每個虛擬硬碟內容將會以區塊（Block）為基本單位分散儲存在多個儲存伺服器上。

1.1 挑戰與突破

DISCO主要有以下三個挑戰與突破：

(1) 高資料取得性

由於DISCO系統採用標準化非特製的硬體（Commodity Hardware）以降低營運成本和避免特殊硬體綁定，然而任何硬體皆有可能損壞，包括了硬碟、硬碟控制器（Disk Controller）、主機板、儲存伺服器、網路卡、網路設備。為了避免資料因為硬體損壞而無法存取，DISCO採用了「N份複本技術（N-way Replication）」將虛擬硬碟的每一個區塊資料都複製N份且儲存在不同的儲存伺服器上。此

技術困難在於維持每一個區塊皆有N份複本和維持每個區塊複本內容的一致性，當使用者寫入資料到虛擬硬碟時，此資料是經由運算伺服器上的DISCO軟體透過網路傳送至N台儲存伺服器上，當網路壅塞或不穩時就可能造成只有部份儲存伺服器收到資料而導致少於N份複本的情況，更嚴重的狀況會發生在要對區塊做覆寫，網路問題造成僅有部份儲存伺服器收到資料，就會造成複本內容不一致的嚴重情況。為了解決這個問題，DISCO設計了「再複製技術（Re-replication）」，DISCO系統有一個模組負責記錄整個系統的元資料（Metadata），其中包括了每一虛擬硬碟區塊所有複本的儲存伺服器位置，當DISCO軟體收到虛擬硬碟區塊寫入請求時，會先查詢元資料來得知複本位置，然後將資料傳送至複本所在的儲存伺服器以作資料寫入；當DISCO軟體收到虛擬硬碟區塊讀取請求時，先查詢元資料來得知複本位置，然後選擇任一複本來取得資料。對於區塊寫入請求，當DISCO軟體傳送資料到複本所在儲存伺服器時，傳送失敗的複本位置將會從元資料中刪除，這可以保證元資料記錄的複本的內容都是一致的，這也用以避免接下來的區塊讀取參考到不正確的複本位置導致讀取了不正確的資料。接者，複本不足的區塊會透過再複製技術在系統背景補足複本數，之後第3章將有更詳細的介紹。

(2) 主要儲存功能和次要儲存功能高度整合

為了解決過往使用者因主要儲存系統（Primary Storage System）與次要儲存系統（Secondary Storage System）廠牌不同而遭遇的耗時繁雜整合作業，DISCO除了提供主要儲存的硬碟讀寫功能外，亦完整整合了次要儲存功能，包括虛擬硬碟快照（Snapshot）、虛擬硬碟回復（Restore）、異地備份與回復。這些功能讓資料在遭遇不同層面的錯誤損壞時都可以得到非常完整的保護。更具體的來說，虛擬硬碟快照功能幫助使用者在人為失誤情況下仍可存取過往時間點的硬碟資料，DISCO讓使用者可自行指定快照頻率和快照保存日期，也可立即建立一個當下快照，亦可隨時將一個快照作回復以存取歷史資料。另外，異地備份回復技術解決了天然災害發生時系統全毀的問題，使用者可部屬多套DISCO系統，然後設定將任何一個DISCO的虛擬硬碟遠端備份到其它的DISCO系統中。

縮短虛擬硬碟快照建立的時間是相當重要且具挑戰的，在建立虛擬硬碟快照時，為了要得到虛擬硬碟當下的資料內容，虛擬硬碟讀寫通常會被暫停，因此，效能越差的快照設計將使得虛擬硬碟讀寫暫停越久，可能導致上層應用程式請求超過預設的等待時間而發生問題。一個直覺的快照設計是在快照建立的當下把虛擬硬碟的元資料和儲存伺服器上的資料全部複製一遍，然而此方法相當耗時沒有效率。

DISCO的快照功能採用了「實體資料區塊共享」、「元資料共享」、和「Redirect on Write」概念，讓快照建立可以不需要複製儲存伺服器上的資料也不需要複製元資料，此零複製的結果使得快照建立可在數秒之內迅速完成。更具體的來說，在DISCO系統中快照也被表示成一個虛擬硬碟，由於快照就是虛擬硬碟在快照執行當下的內容，一個直覺作法是在快照建立時，使用儲存伺服器剩餘空間將虛擬硬碟在儲存伺服器上的所有資料都拷貝一份，為了要省去這個耗時的資料拷貝，DISCO設計讓快照與虛擬硬碟共享儲存伺服器資料，此即「實體資料區塊共享」概念。此概念的一個可行實作，是在快照建立時，先從元資料中找出虛擬硬碟所有區塊的複本位置，由於快照與虛擬硬碟共享這些複本，所以就可透過設定快照區塊的元資料，讓快照區塊的複本位置與相對應虛擬硬碟區塊的複本位置相同，然而此方法需要幫所有快照區塊都設定元資料，為了再省去耗時的元資料複製，DISCO更採用了「元資料共享」概念，由於快照與虛擬硬碟相對應區塊的複本位置相同，所以我們就讓快照與虛擬硬碟共享虛擬硬碟的元資料，也就是說，虛擬硬碟和快照都參照虛擬硬碟的元資料來查看區塊的複本位置。因此在DISCO實作中，快照建立不會設定快照區塊元資料，當要查詢快照區塊的複本位置時，就查詢虛擬硬碟相對應區塊的元資料即可。快照建立後，當收到虛擬硬碟區塊寫

入時，我們將產生新的複本位置讓資料寫至新的實體位置，此即「Redirect on Write」技術，這個是為了要保留原本複本以作為快照區塊內容，以下第4章將對「實體資料區塊共享」、「元資料共享」、和「Redirect on Write」概念和其實作有更詳細的介紹。

(3) 儲存效率優化

儲存系統的儲存效率將決定此系統的擴充能力。大部分使用者習慣在一開始建立虛擬硬碟時指定較大容量，目的在保留未來資料增長的彈性，然而一個沒有完善設計的儲存系統可能就因此只能提供較少的虛擬硬碟數量。

有鑑於此，DISCO透過「自動精簡配置（Thin Provisioning）」和「即時增加儲存伺服器」兩項技術來優化儲存效率。更具體地來說，當使用者建立虛擬硬碟的時候，DISCO並不會真的在儲存伺服器上保留此虛擬硬碟空間，也就是虛擬硬碟建立時不佔用任何實體儲存空間，而是當虛擬硬碟掛載到運算伺服器上且開始作資料寫入時，我們才會配置當下進行寫入的那些區塊的實體位置空間。另外，當剩餘儲存空間不夠的時候，系統管理者可透過「即時增加儲存伺服器」功能讓系統在持續運作下動態的加入新的儲存伺服器。這兩個儲存效率優化技術讓儲存設備投資更有效率，管理者可在初期投資較少經費佈建系統，往後再根據實際資料增長速度而動態增加儲存設備。

此外DISCO亦設計了「去複本技術（De-duplication）」更再提高了儲存效率。此技術將找出系統中內容相同的區塊，重複區塊將被刪除而只留下一份實體區塊空間，並透過資料結構與指標的設計讓這些虛擬硬碟區塊指向同一個實體區塊位址。在此技術中，為了提升區塊內容比對效能我們採用指紋技術（Fingerprint）做第一層過濾，要比對的區塊會先進行指紋的比對，指紋相同的話才會真正進行兩區塊所有位元內容比對，因為指紋的位元數遠小於區塊位元數，因此可提升區塊內容比對效能。

1.2 現況與未來版本規劃

DISCO已完成以上所述功能開發和測試，是一個功能相當完整且穩定的儲存產品。DISCO亦完成了OpenStack Mitaka版本的Cinder驅動程式（Driver）實作，此驅動程式已通過OpenStack官方驗證[13]，因此DISCO更可作為OpenStack系統的儲存方案。另外，更多新功能也在開發測試中，包括抹除碼技術（Erasure Coding）、服務品質保證功能（QoS）、超融合技術（Hyper-Converge），這些功能將在DISCO未來版本中提供。

在以下的章節中，我們將在第二章介紹DISCO系統架構，第三章和第四章分別介紹DISCO提供的主要儲存系統功能和次要儲存系統功能，最後，第五章將呈現實驗結果。

2．DISCO架構

圖2為DISCO系統架構，DISCO軟體模組包括了DISCO-Client、NameNode、DataNode、DSS、WADB-Client和WADB-Server，除了DISCO-Client為C語言Linux核心模組外，其它使用Java程式實做。

DISCO將使用者虛擬硬碟的資料儲存在多個掛滿硬碟的儲存伺服器上，因此每個儲存伺服器都有一個DataNode常駐程式執行於其上負責區塊的讀寫。NameNode程式模組為DISCO核心，記錄了系統所有元資料，其中最重要的是每個虛擬硬碟區塊的儲存伺服器位置，之後的第3.1章節將有更詳細的介紹。此外NameNode也負責了儲存伺服器剩餘可用空間位址記錄、DataNode和DISCO-Client的狀態追蹤、虛擬硬碟的差異區塊記錄（Delta Block Tracking）。DISCO-Client是虛擬硬碟讀寫入口，有安裝DISCO-Client的伺服器才可掛載虛擬硬碟，DISCO-Client會在掛載虛擬硬碟的伺服器上模擬出一個虛擬裝置（Virtual Device），使用者可如同存取實體硬碟般存取此虛擬硬碟，每一個虛擬硬碟的讀寫請求將由Linux Kernel傳送至DISCO-Client做處理，DISCO-Client根據請求的類型和手邊的元資料快取狀況向NameNode傳送實體位置配置請求或元資料查詢請求，最後與DataNode作

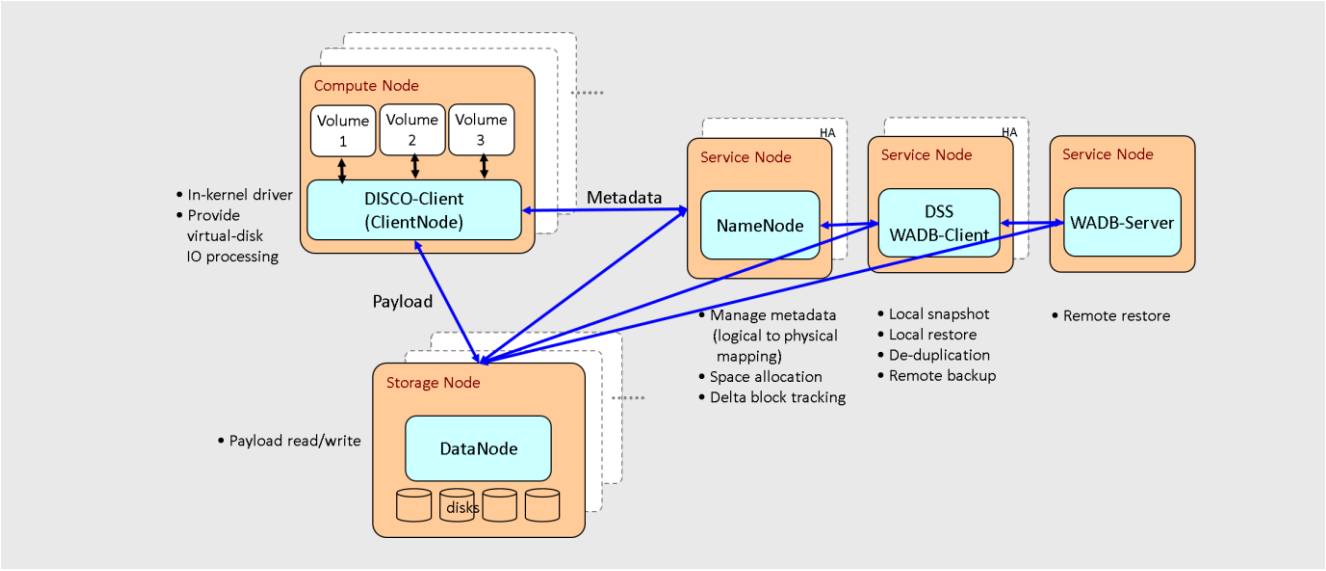

圖 2　DISCO系統架構

資料讀寫交流。

軟體模組DSS、WADB-Client和WADB-Server提供了次要儲存系統功能，DSS主要負責虛擬硬碟快照與回復功能，根據使用者定義的快照頻率去定時執行虛擬硬碟快照，而WADB-Client和WADB-Server模組提供了虛擬硬碟異地快照與回復功能。

由於NameNode和DSS皆儲存了重要的系統元資料，我們設計了HA（High Availability）架構以解決單點故障（Single Point of Failure）問題。如圖3所示，NameNode的HA設計採用主程式主動從屬程式被動（Active Master Passive Slave）架構，我們在兩台相同硬體規格的伺服器上都安裝了NameNode，這兩台伺服器有各自的硬碟且硬碟大小相同。兩個NameNode為主從模式設定，所有的請求只會被傳送到主NameNode做處理，以下為了描述方便，我們稱主NameNode所在的伺服器為主伺服器，而稱從屬NameNode所在的伺服器為從屬伺服器。我們使用Linux-HA公用程式[14]來監視伺服器和NameNode的狀態以及執行錯誤發生時的主從模式切換。更詳細地來說，伺服器的狀態監控是透過兩台伺服器上的Linux-HA工具中的Pacemaker[15]定期傳送心跳（Heartbeat）來達成的，一旦從屬伺服器上的Pacemaker判定主伺服器死亡，Linux-HA將會讓從屬伺服器上的NameNode接管變成新的主NameNode。另外，主伺服器上的Linux-HA會監視其上的NameNode狀態，一旦偵測到突掛會先嘗試將NameNode重新啟動，在一定次數的重啟失敗後，Linux-HA將會讓從屬伺服器上的NameNode接管變成新的主NameNode。

除了狀態監視和主從模式切換設計外，如何在主從模式切換後主NameNode可存取到正確的元資料更是相當的重要。主NameNode會將元資料寫至其下伺服器硬碟，硬碟內容會透過Linux-HA工具的DRBD（Distributed Replicated Block Device）[16]軟體同步到從屬伺服器的硬碟，因此當從屬NameNode接管成為新的主NameNode後，它就可存取其下伺服器硬碟中的元資料以繼續提供服務。為了提高NameNode請求處理速度，我們設計讓請求處理所造成的元資料修改不在請求處理的當下寫至硬碟，而是透過背景執行緒寫至硬碟，此設計可加速請求處理，卻會遭遇到NameNode突掛而記憶體中的元資料修改還來不及寫至硬碟而導致元資料不正確的問題，因此我們設計了修改日誌（Update Log），當NameNode處理一個請求時，相關的元資料修改將首先被記錄在修改日誌中並同步地傳送至從屬NameNode以記錄在記憶體中的修改日誌，接著元資料修改才會被加入到背景執行緒的修改佇列中以執行背景硬碟寫入，當主NameNode突掛而從屬NameNode接管成為新的主NameNode後，它就可根據修改

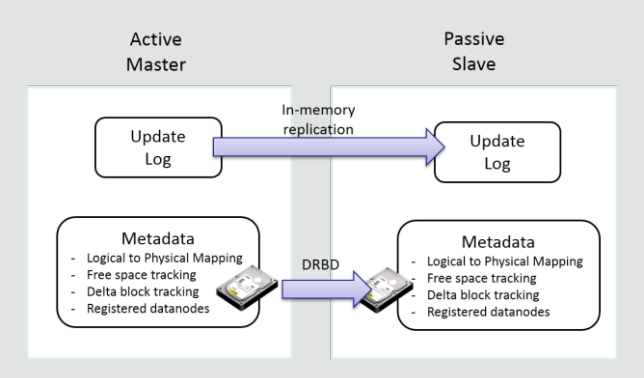

圖 3　DISCO NameNode HA設計

日誌先做硬碟的元資料復原，之後才開放提供服務。

3．主要儲存系統功能

DISCO支援完整虛擬硬碟操作，包括新增（Create）、掛載（Attach）、卸載（Detach）、擴充（Extend）、刪除（Delete）虛擬硬碟。一個虛擬硬碟可掛載到任一裝有DISCO-Client的伺服器上，一次只能掛載到一個伺服器上，需卸載後才可再掛載到其它伺服器使用。擴充功能讓使用者可在虛擬硬碟卸載的情況下擴大虛擬硬碟的容量。

在DISCO系統中，虛擬硬碟是以區塊為基本單位分散儲存在不同的儲存伺服器中，不像GFS[3, 4]和HDFS[5]使用較大的區塊以加速特殊應用的大量資料存取，DISCO的區塊大小預設為較基本且較小的4KB以更廣泛地支援各種企業應用程式、資料庫應用程式和網路應用程式。當使用者操作將一個虛擬硬碟掛載至伺服器時，此台伺服器上的DISCO-Client會在此伺服器模擬出一個虛擬裝置並向Linux Kernel註冊此虛擬裝置，每一個虛擬硬碟區塊讀寫請求將由Linux Kernel傳送至DISCO-Client做處理。在下面的子章節中，我們首先會介紹讀寫請求處理會存取到的元資料，接著是讀寫請求處理流程介紹，最後是讓區塊維持N份複本的Re-replication技術。

3.1 元資料

L2P-Map（Logical to Physical Map）和OW-Flags（Overwritten Flags）是讀寫請求處理時會參考到的元資料。L2P-Map記錄了每個虛擬硬碟區塊在實際儲存伺服器的位置，更詳細的來說，所有虛擬硬碟形成了一個邏輯空間由所有虛擬硬碟的區塊組成，而所有儲存伺服器形成了一個實體空間由所有儲存伺服器的區塊所組成。為了提供高資料取得性我們採用N-way Replication技術，因此每個邏輯空間區塊將有N個複本儲存在N個實體空間區塊位置，因此L2P-Map記錄了每個邏輯空間區塊的N個實體區塊位置。另外，OW-Flags的設計用於記錄一個邏輯區塊是否已寫過資料，邏輯區塊第一次寫入資料時NameNode需要指派其實體區塊位置，之後的資料寫入則覆寫至相同的實體位置即可。因此，在OW-Flags中每個邏輯空間區塊都有一個位元，位元值1代表此區塊曾寫過資料因此要將資料覆寫至原本的實體位置，位元值0代表是第一次寫入因此要分派其實體位置。

3.2 寫入請求處理

圖4描述了虛擬硬碟區塊寫入請求處理流程。DISCO-Client在記憶體中保留了L2P-Map快取，它是NameNode記錄的L2P-Map的一個子集合，用以加速讀寫請求處理，另外，在記憶體中也保留一份OW-Flags。當DISCO-Client收到一個虛擬硬碟區塊寫入請求時，首先會查看OW-Flags以確認是否為第一次寫入，若為第一次寫，就要先向NameNode請求分派實體儲存位置;若非第一次寫，且手邊L2P-Map快取也沒有資料的話，才需要向NameNode查詢元資料。接著，DISCO-Client就傳送資料給N個實體儲存位置的DataNode請求資料寫入，我們設計了Memory ACK和Disk ACK技術以提升處理速度，當DataNode收到DISCO-Client送來的資料時會先傳送一個稱為Memory ACK的確認訊息回去，接著將資料真正寫入至硬碟後會再傳送一個稱為Disk ACK的確認訊息回去，由於每個區塊資料會被傳送至N個儲存伺服器，由於N個伺服器或DataNode同時突掛的機率很低，我們設計讓DISCO-Client在收到N個Memory ACK或1個Disk ACK的情況下就先向Linux Kernel回報請求已做完，如此便可提高請求處理速度，接著DISCO-Client仍會在背景繼續接收剩下的所有ACK，在一定時間後沒有收到Disk ACK的實體位置則會被判定成失敗，然後通知NameNode以做L2P-Map元資料修改。

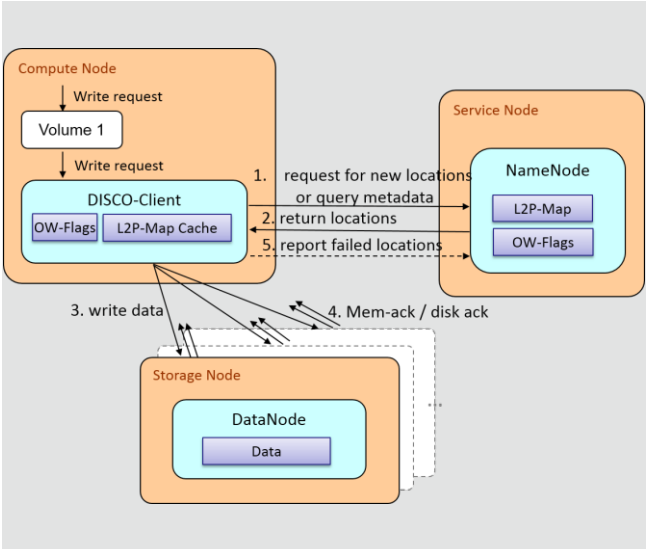

圖 4　虛擬硬碟區塊寫入請求處理

3.3 讀取請求處理

圖5描述了虛擬硬碟區塊讀取請求處理流程。當DISCO-Client收到一個區塊讀取請求時，會先查詢手邊的L2P-Map快取，若沒有資料的話才會向NameNode查詢，接著，向其中一個實體位置的DataNode送出讀取請求，若DataNode成功回傳資料則DISCO-Client馬上回傳資料給Linux Kernel，若一定時間內沒有收到回應就會嘗試傳送讀取請求給其它實體位置的DataNode，最終，DISCO會將曾傳送請求但沒有收到回應的位置判定成失敗，然後向NameNode回報以做L2P-Map元資料的修改。

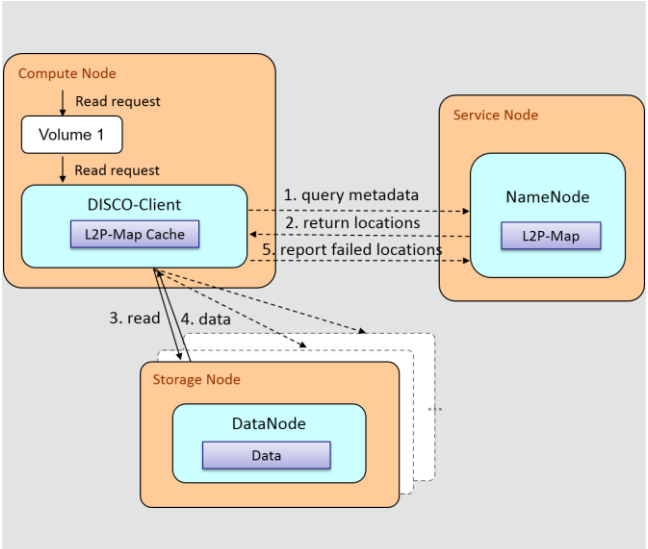

圖 5　虛擬硬碟區塊讀取請求處理

3.4 Re-replication技術

隨時保持每個區塊都有N份複本是相當重要的，如以上章節3.2和章節3.3所述，在讀寫實體位置發生問題的時候，DISCO-Client會通報NameNode修改L2P-Map將有問題的實體位置移除，目的讓實際寫入狀況和元資料是一致的，此步驟對於寫入失敗的情況更是重要，因為若沒有將失敗實體位置從元資料中移除，之後的區塊讀取可能就從錯誤的實體位置存取到不正確的資料了。另外，DISCO實作了Re-replication技術在背景幫區塊補足不足的複本數，當DISCO-Client向NameNode回報讀寫失敗時，所有不足複本數的區塊將被記錄起來，一個背景執行緒將負責複本數補足的工作，它會先找到一個新的實體位置，然後傳送複製請求給有此區塊資料的DataNode請它將資料讀出後傳送至新實體位置DataNode做寫入，如此就完成複本數補足的工作。

4．次要儲存系統

DISCO提供的次要儲存系統功能包括了虛擬硬碟快照與回復，以及虛擬硬碟異地備份與回復。使用者可執行虛擬硬碟的當下立即快照，或設定讓系統自動定期建立快照，異地備份功能也有一樣的設定。

4.1 快照

我們透過實體資料區塊共享、元資料共享、和Redirect on Write技術讓快照建立可在數秒之內完成，且達到資料與元資料零複製的突破。由於快照就是虛擬硬碟所有邏輯區塊在某一個瞬間的內容，因此在DISCO系統中快照也被表示成一個虛擬硬碟，也需要元資料記錄它所有邏輯區塊的實體區塊位置。一個可達到資料零複製的想法，是讓快照與虛擬硬碟共享實體區塊，因此快照建立的當下快照與虛擬硬碟每個相對應的邏輯區塊就指向相同的實體區塊位置，此為實體資料區塊共享概念。快照建立之後，虛擬硬碟邏輯區塊的第一次寫入將被寫至新的實體區塊位置，以保留舊實體區塊資料給快照，此為Redirect on Write技術。為了達到實體資料區塊共享，一個可行方法是在快照建

立當下，先從L2P-Map讀出虛擬硬碟每個邏輯區塊的實體區塊位置，接著在L2P-Map中建立快照的元資料以設定其邏輯區塊指向虛擬硬碟相對應邏輯區塊的實體區塊位置，但此方法需要複製虛擬硬碟所有邏輯區塊的L2P-Map元資料。有別於此，DISCO讓快照與虛擬硬碟也共享元資料，因此可再省去元資料複製的過程，此為元資料共享概念。

具體來說，DISCO透過「虛擬硬碟父子關係」與「Valid Bitmap」兩資料結構來實作元資料共享概念。當兩虛擬硬碟有父子關係時，兩虛擬硬碟就可共享L2P-Map資料，當查詢小孩虛擬硬碟一個邏輯區塊的實體區塊位址時，若它在L2P-Map有記錄則以此記錄為主，但若沒有記錄但爸爸虛擬硬碟相對邏輯區塊在L2P-Map有記錄的話，爸爸記錄的實體區塊位置就是小孩的實體區塊位置。另外，所有虛擬硬碟邏輯區塊在Valid Bitmap中將有一個位元以表示它在L2P-Map是否有記錄，位元數值0代表在L2P-Map沒有記錄。由於DISCO將快照也表示成一個虛擬硬碟，在建立虛擬硬碟快照時，我們首先設定快照與虛擬硬碟父子關係使得虛擬硬碟為父而快照為子，也設定快照所有邏輯區塊的Valid Bitmap位元為0，並將虛擬硬碟所有邏輯區塊的OW-Flags全部重設為0，此即完成了快照建立，這也代表在此當下快照邏輯區塊的實體位址全部參考虛擬硬碟邏輯區塊在L2P-Map的記錄，因此快照與虛擬硬碟就共享了實體區塊與L2P-Map元資料。

如以上所述，在快照建立時虛擬硬碟所有邏輯區塊的OW-Flags全部重設為0，因此快照建立之後，當DISCO-Client收到虛擬硬碟區塊的寫入請求時，由於OW-Flags的位元已被設成0所以就會向NameNode請求分派新的實體區塊位址讓資料轉而寫入新的位置，當NameNode收到新位置請求時，會先將要作寫入的邏輯區塊的L2P-Map記錄先轉而設定給快照相對應的邏輯區塊，並把快照相對應邏輯區塊在Valid Bitmap中的位元設成1，接著才會修改虛擬硬碟區塊的L2P-Map記錄以指向新的實體區塊位置，從此快照跟虛擬硬碟的這個邏輯區塊就不再共享L2P-Map記錄。

4.2 異地備份

虛擬硬碟異地備份功能讓使用者可將一套DISCO中的虛擬硬碟資料備份到另一套DISCO系統。圖6描述了異地備份流程，此圖假設一套DISCO安裝在台北，一套DISCO安裝在新竹，當使用者設定將台北DISCO的Volume1備份至新竹DISCO後，第一次執行異地備份時我們會先在新竹端產生一個相對應的虛擬硬碟，接著對台北端Volume1建立一個快照，然後將此快照內容全部傳到新竹端並寫入新竹端的虛擬硬碟，最後再建立新竹端硬碟的快照，此即完成了第一次的異地備份。之後每次異地備份也都會先在台北端建立快照，資料傳送並寫入到新竹端硬碟後再建立相對應的新竹端快照。每次異地備份時，我們只傳送此次快照與前次快照的資料差異（Delta）以提高異地備份效能。另外，使用者可以選擇使用網路或實體硬碟兩種不同途徑做為兩DISCO系統間資料傳送的媒介，以實體硬碟做傳輸可解決兩系統間廣域網路頻寬不足的問題。

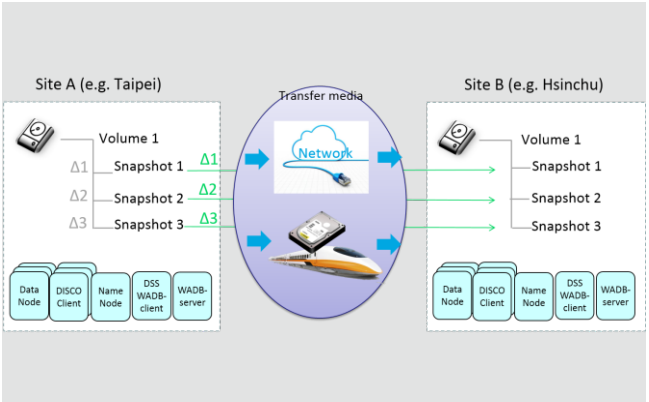

圖 6　虛擬硬碟異地備份

5．實驗結果

為了了解DISCO效能，此章節中我們比較了DISCO和Ceph[10, 11]的資料讀取效能，Ceph是一個開源（Open Source）儲存系統，除了提供區塊儲存服務外亦提供了物件（Object）和檔案（File）儲存服務，目前被廣泛使用在很多OpenStack平台中用以提供區塊儲存服務。DISCO測試環境使用兩個儲存伺服器，每個儲存伺服器有5個透過磁碟陣列卡

（Hardware Raid Card）組成的8TB RAID5硬碟以提供DataNode做資料儲存。為了對等比較，Ceph測試環境也佈署了兩個OSD軟體各執行於一個儲存伺服器上，每個儲存伺服器也一樣有5個由磁碟陣列卡組成的8TB RAID5硬碟，且格式化成xfs檔案系統以供OSD儲存資料，另外有5個2TB RAID1硬碟以供Ceph的OSD軟體模組作5個資料碟的Journal之用。兩個測試環境中的所有伺服器都是用10G Ethernet網路卡連接，且每台伺服器記憶體皆至少16GB。DISCO和Ceph皆設定成Two-way Replication。Ceph使用INFERNALIS (9.2.0)版本做測試。

5.1 循序讀取效能

我們使用Linux dd工具來測試單一虛擬硬碟循序讀取（Sequential Read）的吞吐量（Throughput），每次dd測試的讀取資料總量為6GB。我們分別測試兩種情況：Cache Miss和Cache Hit。對於Cache Miss測試項目，在測試前我們會先清除運算伺服器和儲存伺服器上Linux檔案系統的快取，目的是測試軟體系統真正從儲存伺服器上讀取資料的效能。相反的，Cache Hit的測試則沒有清除運算伺服器和儲存伺服器上Linux檔案系統的快取，因此每次測試前我們會先使用dd工具將要測試的區塊讀取一遍，因此接下來測試過程要被讀取的區塊的資料應大多會在Linux快取中。圖7和圖8分別為Cache Hit和Cache Miss的測試結果，橫軸為dd的bs(Block Size）參數設定。由測試結果看來，在不同的dd Block Size情況下，DISCO循序讀取吞吐量皆優於Ceph。DISCO-Client在虛擬硬碟掛載於運算伺服器上時，透過Linux的blockdev指令的setra參數增大了虛擬硬碟的Read Ahead大小，另外DISCO-Client設計了Request Merge技術讓多個連續邏輯區塊的讀取可以在單一請求中處理完成，如此縮減了請求延遲時間（Latency）也提高了吞吐量。

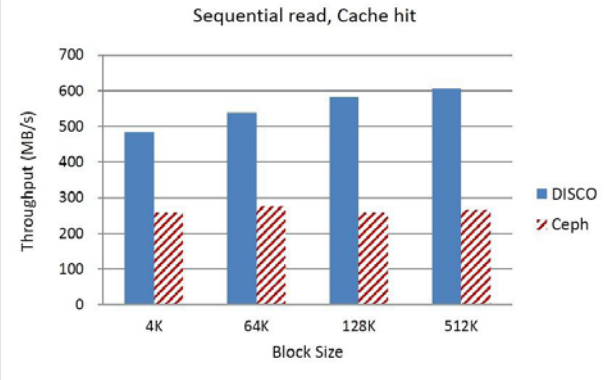

圖 7　Cache hit情況的循序讀取效能

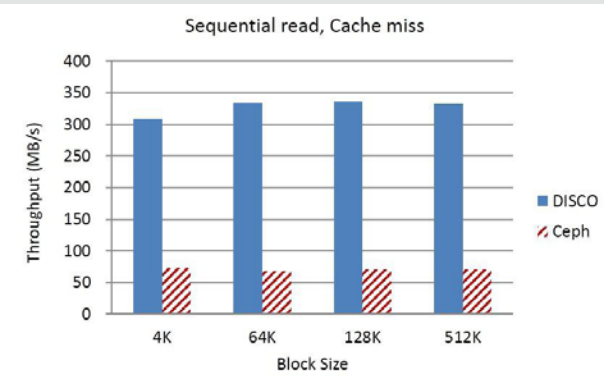

圖 8　Cache Miss情況的循序讀取效能

5.2 隨機讀取效能

我們使用fio工具來測試單一虛擬硬碟隨機讀取（Random Read）效能，我們各在DISCO和Ceph系統產生一個6GB的虛擬硬碟，每一次fio測試時間設定為2分鐘。圖9和圖10分別為Cache Hit和Cache Miss情況的測試結果，在大部份的fio Block Size設定下，DISCO隨機讀取吞吐量皆低於Ceph。我們觀察到在fio測試時，Ceph儲存伺服器上的5個資料硬碟大多會被同時存取，而DISCO環境每次大約只有3個資料硬碟同時被存取。此原因在於DISCO的設計中虛擬硬碟將以2GB為單位做實體區塊位置分配，因此6GB虛擬硬碟資料就大約落在3組2GB實體位置，因此也造成隨機存取情況下最多同時只有3個儲存伺服器的資料硬碟被存取；相反的，Ceph採用較小的MB等級單位作虛擬硬碟切割和儲存，所以在隨機讀取的測試下，就會有更多儲存伺服器的資料硬碟被平行存取，自然吞吐量就較高。DISCO正在實做DataNode Load

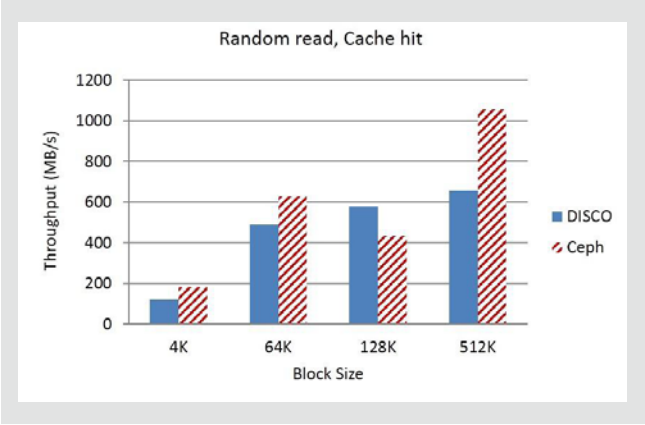

圖 9　Cache Hit情況的隨機讀取效能

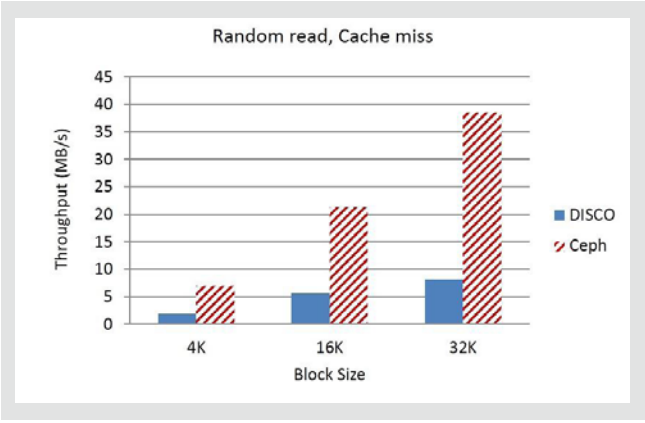

圖 10　Cache Miss情況的隨機讀取效能

Balance技術，由於每個邏輯區塊都有N份複本，因此可在讀取請求時選擇較輕負載的DataNode做讀取，即可讓多台DataNode平行處理所有虛擬硬碟的請求以提升總吞吐量。

6．結論

第一版DISCO已經完成功能驗證與測試，此版本提供了齊備的虛擬硬碟操作和資料備份功能讓資料在各種錯誤與災害情況下都可受到完整保護，多項技術突破和實作更大大的提升了儲存效率，使用者更可透過與OpenStack Mitaka版本相容的Cinder Driver將DISCO作為OpenStack平台的儲存系統。更多新功能也在開發測試中，包括抹除碼技術、服務品質保證功能、超融合技術，這些功能將在DISCO接下來的版本提供。

參考文獻

[1] “Cisco分析全球2018年雲端數據規模將為傳統數據的3倍,“ 科技政策研究與資訊中心，2015年1月20日，http://iknow.stpi.narl.org.tw/post/Read.aspx?PostID=10584

[2] "Cisco Global Cloud Index: Forecast and Methodology, 2013-2018," Cisco, 2014.

[3] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," in 19th ACM Symposium on Operating Systems Principles, 2003.

[4] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in Globally Distributed Storage Systems," in 9th USENIX Symposium Operating Systems Design and Implementation, 2010.

[5] Hadoop Distributed File System (HDFS), https://hadoop.apache.org/docs/r2.7.1/

[6] "An Introduction to Gluster Architecture," Gluster Inc., Whilte Paper, 2011.

[7] "Performance in a Gluster System," https://s3.amazonaws.com/aws001/guidedtrek/Performance in a Gluster Systemv6F.pdf

[8] "Lustre: A Scalable, High-Performance File System," Cluster File Systems, Inc., White Paper, 2002.

[9] P. J. Braam, "The Lustre Storage Architecture," Cluster File Systems, Inc., While Paper, 2003.

[10] S. A. Weil. "Ceph: Reliable, Scalable, and High-Performance Distributed Storage," Ph.D. thesis, University of California, Santa Cruz, 2007.

[11] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn. "Ceph: A Scalable, High-Performance Distributed File System," in 7th Symposium on Operating Systems Design and Implementation, 2006.

[12] Sheepdog,

https://sheepdog.github.io/sheepdog/
[13] OpenStack Block Storage (aka Cinder) Drivers: https://wiki.openstack.org/wiki/CinderSupportMatrix
[14] Linux-HA, http://www.linux-ha.org/wiki/MainPage
[15] Pacemaker, http://clusterlabs.org/
[16] DRBD, http://drbd.linbit.com/

作者簡介

朱怡虹

現任職於工研院資通所資料中心系統軟體組擔任資深工程師，專長為儲存系統設計、和資料探勘技術。
E-mail: vicki.chu@itri.org.tw