東海大學

資訊工程研究所

碩士論文

指導教授: 林祝興博士

雲端及 GPU 技術對於破解雜湊函數 SHA-1 的效能研究

On the Performance of Cracking Hash Function SHA-1 Using Cloud and GPU Technologies

研究生: 朱田彬

中華民國 106 年 7 月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 朱 田 彬 所提之論文

雲端及 CPU 技術對於破解雜湊函數 SHA-1 的效能研究

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召 集 人 _____ 簽章

委 員 _____

_____

_____

指 導 教 授 _____ 簽章

中華民國 106 年 7 月 14 日

摘　要

物聯網（IoT）的發展改變了傳統的網路架構。然而，安全問題已經成為佈署 IoT 的主要障礙，IoT 通常採用雜湊函數進行安全認證。由國家標準技術研究所提出的安全雜湊演算法（SHA）是一系列雜湊函數，包括 SHA-0，SHA-1，SHA-2 和 SHA-3。SHA 可用於計算產生固定長度訊息輸出，稱為訊息摘要。SHA-1 不是可逆的，已廣泛應用於安全應用和協議，包括 TLS 和 SSL。本研究通過使用雲端運算和 GPU 技術計算雜湊並與目標雜湊值進行匹配來找到密碼的碰撞。對於雲端運算系統，我們使用由四個實體計算機組成的 Hadoop 分佈式系統來實現匹配演算法，一個用於主節點，另外三個用於工作節點。此外，MapReduce 用於節點處理大量操作。對於 GPU 平行運算系統，我們使用數千個 GPU 執行緒對 CPU 分配的數據進行平行運算。我們進行實驗來驗證 SHA-1 訊息摘要碰撞的產生是否可行，並指出在新興雲端運算和 GPU 平行化時代 SHA-1 是否仍然足夠對抗對手。

關鍵字: 雲端運算、hadoop、GPU、資訊安全、SHA-1、雜湊函數

Abstract

The development of Internet of Things (IoT) has changed traditional ideas of networks. However, security issues have become major obstacles to the deployment of IoT, which commonly adopts hash functions for security authentication. The Secure Hash Algorithm (SHA), published by the National Institute of Standards and Technology, is a family of cryptographic hash functions, including SHA-0, SHA-1, SHA-2, and SHA-3. SHA can be used to compute fixed-length digits called message digests for digital messages. SHA-1 is not invertible and has been widely used in security applications and protocols, including TLS and SSL. This study cracks code words by matching them with target hash values using cloud computing and GPU technologies. For the cloud computing system, we implement the matching algorithm using a Hadoop distributed system consisting of four physical computers, one used for the master node and the other three used for worker nodes. Also, MapReduce is used in nodes to handle large amounts of operations. For the GPU parallel computing system, we use thousands of GPU threads to operate in parallel on the data allocated by the CPU. We perform experiments to verify whether it is feasible to crack SHA-1 message digests, and indicate whether SHA-1 is still secure enough against opponents in the emerging cloud computing and GPU parallelization era.

Keywords: cloud computing; Hadoop; GPU computing; IoT security; SHA-1; hash function

致謝

兩年研究所的日子轉眼間就要結束了，這些日子以來有許許多的人們需要感謝，感謝有你們的陪伴，幫忙與諒解。

首先需要感謝的是在大學時就幫助我許多，從大學專題到研究所的指導老師 林祝興老師，願意讓我在研究所時嘗試這麼特殊的議題，真的非常的感謝老師一路上的包容與幫助。

朱田彬 於東海大學研究所 資安實驗室　106 年 7 月

目錄

摘要 **I**

Abstract **II**

目錄 **III**
圖目錄 IV
表目錄 V
第一章 簡介 1
第二章 背景知識 3
2.1 Hadoop 3
2.2 GPU 平行運算 5
2.3 SHA-1 8
2.4 生日攻擊法 9
第三章 演算法設計 11
第四章 實驗結果 15
4.1 實驗環境 15
4.2 實驗 16
4.2.1 雲端環境中單一節點不同數據量下的運行速度的研究 16
4.2.2 雲端環境下不同節點數的效能比較 17
4.2.3 雲端環境下不同節點數破解不同字元數所需的時間研究 20
4.2.4 GPU 相同執行緒不同數據量執行時間研究 21
4.2.5 GPU 不同執行緒數量的效能比較 21
4.2.6 GPU 不同執行緒數量與破解不同字元數的執行時間研究 22
4.2.7 雲端系統與 GPU 平行運算的加速比 23
第五章 結論 26
參考文獻 27

圖目錄

2.1 MapReduce 運作流程 5
2.2 CUDA 主要架構圖.....7
2.3 MIMD VS SIMD 7
2.4 SHA-1 雜湊演算法流程圖 9

3.1 平行運算破解 SHA-1 流程圖 13
3.2 單一節點（或執行緒）上的簡單雜湊值碰撞演算法的虛擬碼 14

4.1 雲端環境單一節點每次分配不同數據量的運行速度研究，它們分 別是: (a) 十億到百億次 (b) 一億到十億次 (c) 一千萬到一億次 (d) 一百萬到一千萬次 17
4.2 雲端環境不同節點數的效能比較.....18
4.3 雲端環境不同節點數的加速比.....19
4.4 雲端環境下不同節點數破解 6 個字元數所需的時間..... 20
4.5 雲端環境下不同節點數破解不同字元數所需的時間.....20
4.6 GPU 1024 執行緒破解 10 的 4 次到 10 的 10 次所需時間，(b) 將 時間取對數後的折線圖 21
4.7 GPU 不同數量的執行緒間的效能折線圖 22
4.8 GPU 不同數量的執行緒破解不同字元數的時間折線圖 23
4.9 單機與雲端的加速比 24
4.10 單機與 GPU 的加速比.....25

表目錄

4.1 實驗環境 15
4.2 雲端環境中單一節點不同數據量下的運行時間 (十億到一百億次) (B: 十億) 16
4.3 雲端環境中單一節點不同數據量下的運行時間 (一億到十億次) (HM: 一億) 16
4.4 雲端環境中單一節點不同數據量下的運行時間 (一千萬到一億次) (TM: 千萬)..... 16
4.5 雲端環境中單一節點不同數據量下的運行時間 (一百萬到一千萬次)(M: 百萬) 17
4.6 雲端環境不同節點數的加速比 (TM: 千萬) 18
4.7 GPU 不同執行緒數量的執行時間.....21
4.8 GPU 不同執行緒數量破解不同字元數的執行時間 22
4.9 雲端環境、GPU 與單機的加速比.....24

第 一章　簡介

物聯網（IoT）的發展改變了傳統的網路觀念。無線傳感器在這方面發揮了重要作用，但安全問題成為佈署 IoT 的主要障礙 [1,2]。目前，雜湊函數通常使用於 IoT 的安全認證過程，雜湊函數的安全會嚴重影響 IoT 的資訊安全與隱私保護。

SHA-1 是一個已經被廣泛使用的雜湊函數，可以將任意長度的訊息輸入經由演算得到一組 160bit 的訊息摘要，因為要找到兩組不同訊息使具有相同訊息摘要或者藉由訊息摘要還原訊息是非常困難的，故常用來驗證資料的完整性與保護密碼的儲存等用途。近年來，由於電腦技術的飛速發展，使得原本需要長時間的運算變得更加快速，SHA-1 的安全性受到嚴重質疑 [3-7]。不少攻擊 SHA-1 的演算法被提出，例如，生日攻擊演算法，此外，雲端運算和圖形處理單元（GPU）的開發也使得耗時的操作變得更容易處理 [8-12]。

雲端運算的進步不僅能夠實現單個節點的更好的性能，而且還允許雲端伺服器通過雲端系統的快速部署快速提供許多運算節點的服務。例如，Google 和其他大型雲端服務提供商均有超過一百萬台伺服器提供服務。這些都使得使用者更容易取得或租借更多的節點來進行運算，讓雲端的成本更低。

GPU 平行運算是由 NVIDIA 所提出的名為 CUDA 的一種整合技術，透過這項技術可以使用 GPU 來進行平行運算，GPU 是一種特殊的處理器，相較於 CPU 可以應用於各種通用運算，GPU 具有數百到數千個核心，在處理重複並簡單的多重運算上，經過最佳化，在某些運算上比 CPU 執行相同的運算速度快 10 到 100 倍。

在本文中，為了研究雜湊值碰撞的時間成本，即找到 2 組不同訊息擁有相同 SHA-1 的訊息摘要，並推斷所需的資源和性能，我們實現雲端分佈式運算和 GPU 平行運算系統來對雜湊值進行解碼由 7 個或更少字符的組合組成的文字。最後，我們將展示雲端運算的性能加速率（或加速比）以及 GPU 平行運算方法超過獨立 PC 的速度。

本論文一共分為五章: 本章為簡介，說明研究動機與目標；第二章中討論背景相關知識與現有的文獻探討；第三章主要是介紹實驗中所使用的演算法；第四章為實驗環境與在雲端環境和 GPU 中的效能探討；第五章則是結論以及未來展望。

第 二章　背景知識

本章節介紹了所使用的雲端運算平台，即 Apache Hadoop，GPU 平行計算和安全雜湊演算法 SHA-1。

2.1 Hadoop

Apache Hadoop 作為開源軟體，是由 Google 公司所發表的為大數據提供分佈式處理、儲存和分析框架; 除了基礎框架之外，它還包括其他搭配軟體，如 Apache Spark、HBase、Storm、Pig、Hive、Sqoop、Oozie 和 Ambari [13-15]。Hadoop 是一個可以連接數千台伺服器作為超級計算機的叢集系統。Hadoop 分為兩個部分，第一個部分是儲存部分，它採用 Hadoop 分佈式檔案系統（HDFS）進行資料存儲。在 HDFS 中，將儲存的資料分散儲存在 DataNodes，所以可以使用多達數千個節點來存儲資料，將分散的儲存資料整合成一個具有容錯並有高效率跟超大容量的儲存環境。HDFS 是屬於"Write-once-read-many" 的模式，每筆資料一次寫入，多次存取，該模式可以降低突發性的控制要求所產生的衝突，提高資料整合與多次存取的效能。在 Hadoop 系統中巨量資料和運算產生的暫存檔案都是存放在 HDFS 中。Hadoop 主要是由 Master/Slave 架構組合而成，HDFS 是由 NameNode 以及 DataNode 組成。HDFS 的運作是，叢集系統中數以千計的節點用於存放資料，系統中會有一個節點為 Master Node 且同時也是 NameNode，為 HDFS 的核心元件，管理整個 HDFS 檔案讀取、寫入... 等 操作。整個 Hadoop 系統中只有一個 NameNode，其餘節點稱為 Slave Nodes 或 DataNodes，為 HDFS 的資料儲存節點。而儲存資料的方式是，先將它分割成 數個區塊 (block)，並將文件分割成相同大小的區塊（通常為 64 MB），並將每

個區塊複製成 3 份副本並分配到 DataNodes 中進行儲存; 而 NameNode 用於監視儲存副本的狀態，並從其他 DataNodes 進行資料複製，以使任何丟失或損壞的副本在系統中始終具有 3 個完整的副本。所以雖然在使用者介面上所示只有一個檔案，但這個檔案是分散在不同的節點中儲存。

第二個部分是處理部分，用於大規模數據集的並行運算。此軟體架構是雲端運算的關鍵技術，將要執行的問題，拆解成 Map 和 Reduce 的方式來執行，以達到分散運算的效果。開發者可以透過簡單的撰寫程式，利用大量的運算資源，更加快速的處理巨量資料。MapReduce 類似於演算法中的分治方法，將複雜問題分解為較小的問題，並遞歸解決，然後將解決方案組合成單個結 果，Map 和 reduce 的概念已被廣泛應用於程式語言。MapReduce 分為 Map 和 Reduce 兩個階段，通過分佈式運算技術處理跨節點的巨量資料。在 Map 階段，分佈在節點之間的資料通過 Map 任務進行過濾和分類，以生成中間資料; 之後，在 reduce 階段，通過減少任務並收集和總結節點間的運算結果，以獲得最終結 果。這樣，大量資料可以由數千個伺服器同時處理，大大減少資料處理時間。Figure 2.1 是 MapReduce 運作的流程圖。

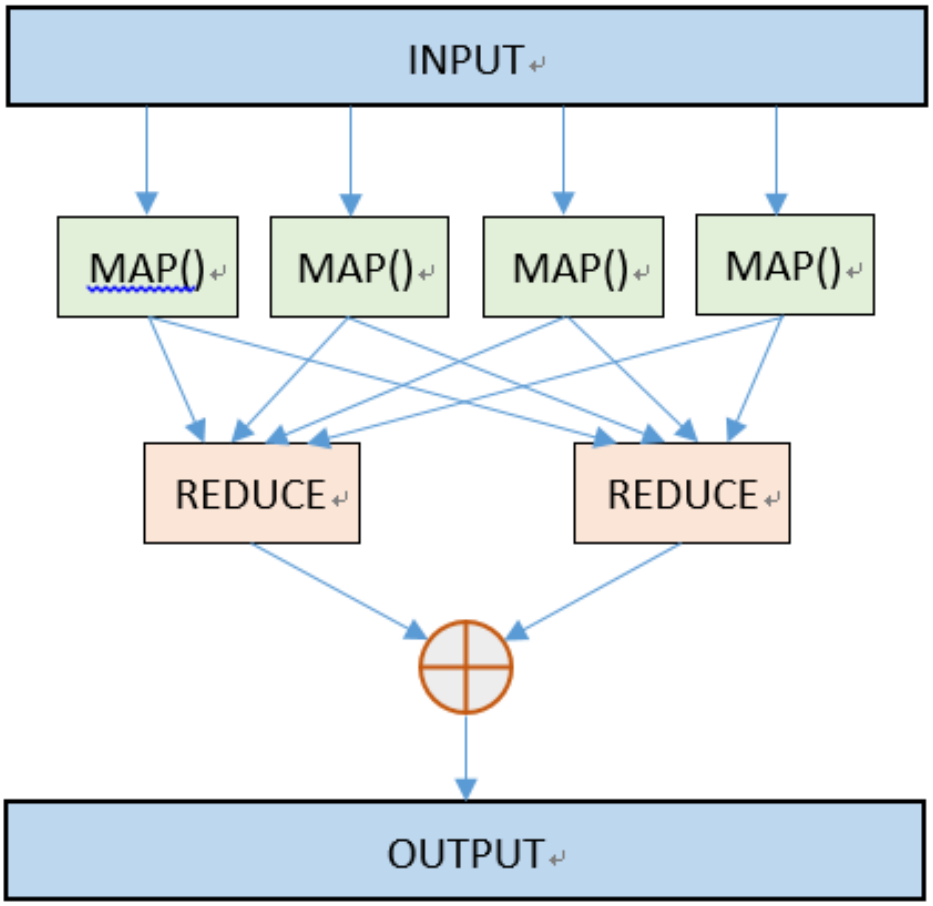


Figure 2.1: MapReduce 運作流程

2.2 GPU 平行運算

過去 GPU 被製造出來的目的主要是為了處理圖形的渲染以及顯示，而近年來 GPU 漸漸的被製造成除了圖形顯示之外還可以進行科學運算的裝置，最早被稱為 GPGPU(General-purpose computing on graphics processing units)。由於 GPGPU 的出現讓 GPU 擁有了科學運算的能力，也就是我們可以在 GPU 上進行程式編寫 [16-18]。而後出現了 OpenCL 以及 CUDA 這兩個用於進行 GPU 程式編寫的 API，其中的差異在於 CUDA 只能用於 NVIDIA 公司旗下的產品。除了 API 的差異以外，GPU 的硬體也跟過去大不相同。GPU 平行運算與 CPU 平行運算的差異在於 GPU 除了擁有 CPU 無法比擬的大量核心數之外，GPU 的架

構是屬於 SIMD(Single Instruction Multiple Data) 的架構 [19]。這代表了 GPU 在科學計算上一次只能運行一樣工作，雖然擁有大量的核心數量，但同一時間只能處理一件事情，無法像 CPU 那樣可以進行多工的分配。所以可以知道適合 GPU 的運算通常都是單純且大量的計算，並且 CPU 在 GPU 進行運算的同時仍然可以擁有處理運算的能力。這讓我們的演算法除了單純的拆分成 SIMD 架構的平行化之外，更可以利用 CPU 以及 GPU 所組成的 HSA(Heterogeneous System Architecture) 架構進行更進一步的優化 [20]。

CUDA 是 NVIDIA 的平行運算架構，可運用繪圖處理單元 (GPU) 的強大處理能力，大幅增加運算效能。通過 CPU 使 GPU 中的數千個執行緒執行一樣的計算，將原本冗長且單一的計算分割成數千個部分，由 GPU 執行緒進行運算，複雜的部份由 CPU 處理，使得長時間的運算得以更快得到結果。雖然 GPU 擁有數千個執行緒，但是過於複雜的任務反而運算效率會低於 CPU，例如，GPU 對於邏輯判斷容易導致效率降低，對於 GPU 來說同一個 Warp 內的 32 個執行緒如果判斷出 2 個不同的分支，則這一個 Warp 的執行緒將會 2 個分支都運行，使得運行時間增加且判斷容易出錯。CUDA 的主要架構如 Figure 2.2。

對於 GPU 的演算架構-SIMD， 可以與 CPU 的 MIMD 架構進行比較如 Figure2.3。我們可以發現 SIMD 的架構一次只能處理一種運算但 MIMD 可以一次分別處理各種不同的運算，所以這種架構對於演算法來說必須要經過一定程度的改良才能在 GPU 上有著顯著的效益。由於 GPU 的架構為單指令多資料 (SIMD) 架構，所以在演算法上必須有著資料間的高獨立性以及高平行度，並且大量重複計算的運算必須要是簡單的，並且盡量不要放置太多邏輯處理的方法，不然在 GPU 的平行運算上所產生的分支運算將會拖垮整體的效能。

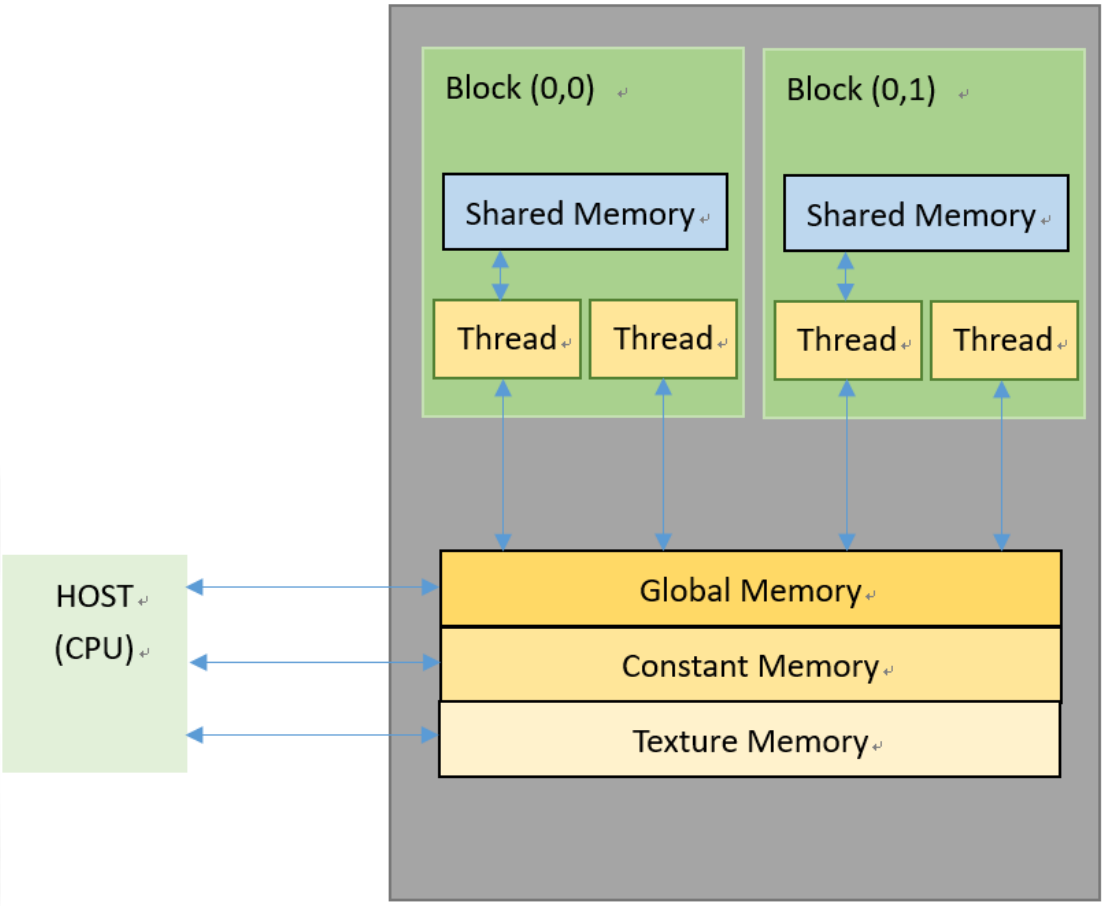


Figure 2.2: CUDA 主要架構圖

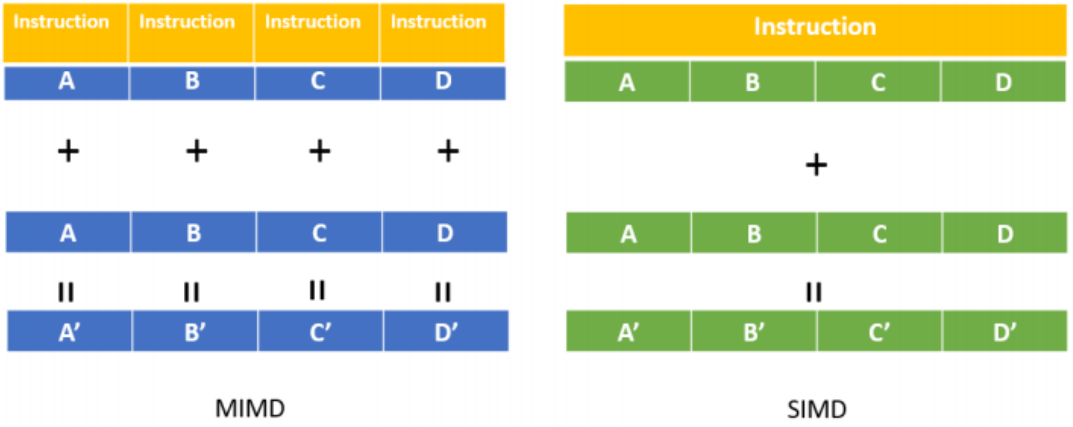


Figure 2.3: MIMD VS SIMD

2.3 SHA-1

SHA-1 是由美國國家標準與技術局（NIST） 發布的聯邦訊息處理標準（FIPS）出版物中批准的安全雜湊演算法。SHA-1 可以輸入任意長度的訊息經過運算而產生輸出長度為 160 位的訊息摘要。輸入訊息的長度可以小於 512 位; 然而，當它大於 512 位時，訊息被分成幾個訊息區塊。原始訊息摘要和新訊息區塊用於生成新的訊息摘要。雜湊函數是一個不可逆的單向函數。這種演算法是安全的，因為通過數學運算來破譯訊息摘要是非常困難的。雜湊函數具有非常強的雪崩效應，即，訊息的微小變化將引起訊息摘要的劇烈變化。

目前，SHA-1 是最流行的安全雜湊演算法，然而，它面臨著嚴重的安全挑戰 [5,6]。通過強大的雲端運算手段和 GPU 等高級硬體，SHA-1 發現碰撞的時間大大減少 [3]。越來越多的研究人員正在尋找增加 SHA-1 抵抗碰撞的抗性的方法 [4]。在本文中，我們主要研究使用下一節描述的簡單匹配演算法以找到 SHA-1 碰撞所需的時間和資源。

SHA-1 的運作流程分為 2 個部分，第 1 個部分為將訊息切割成 512bit 區塊，不足則補 0，並將分好的訊息再分為 32bit 一組共 16 個的陣列，依特定公式擴展成 80 個的陣列。第 2 個部分則是執行 Figure2.4 的函數，並迭代 80 次，每次迭代均代入訊息陣列中的一個單位在 Wt。其中 A,B,C,D,E 是 32bit 的狀態 (state)，F 為一非線性的函數，< < <n 則是向左位移 n 個 bit，K 則是依根據迴圈數代入不同的常數。

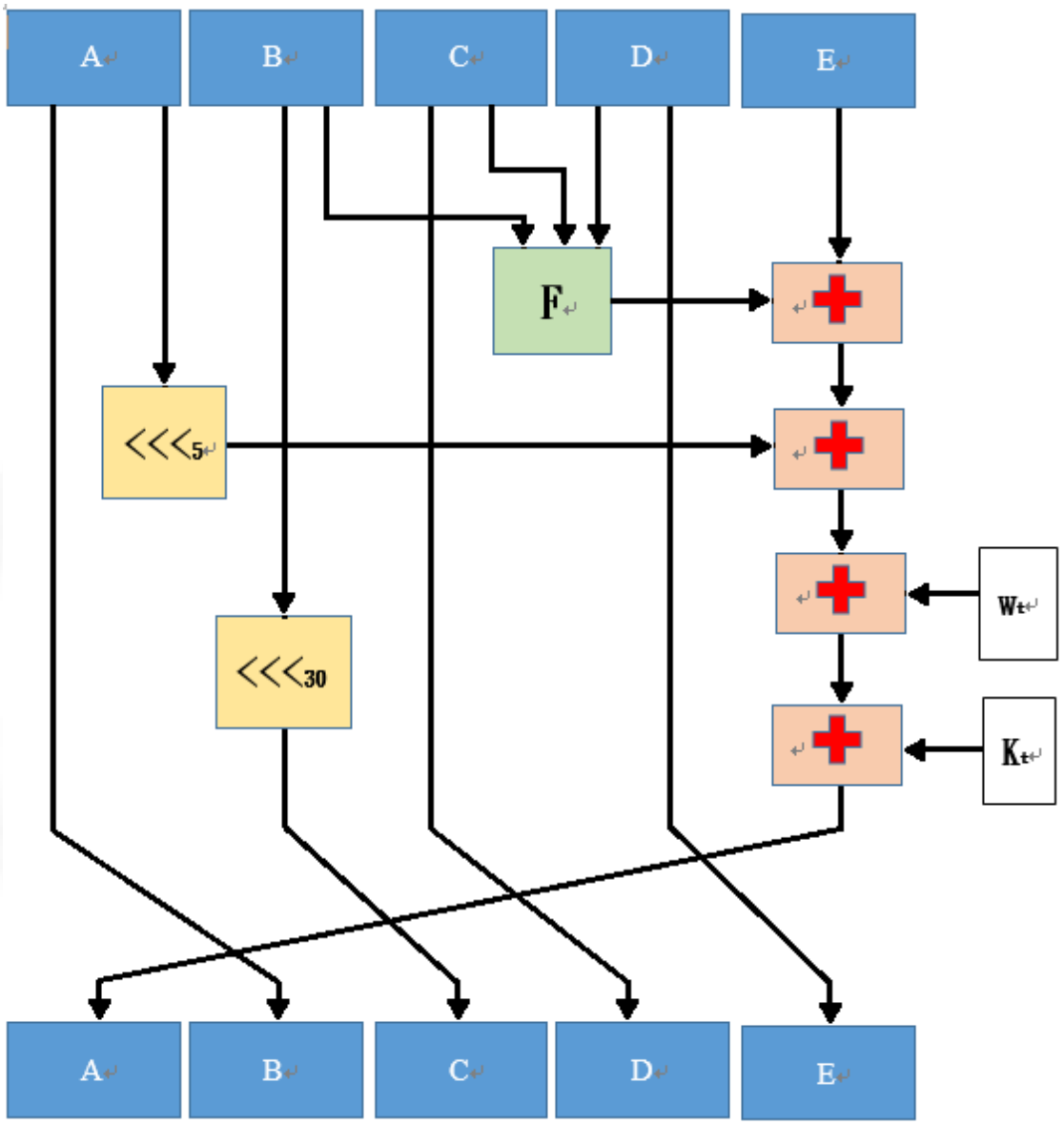


Figure 2.4: SHA-1 雜湊演算法流程圖

2.4 生日攻擊法

生日攻擊法是一種經常運用在密碼破解的方法，而生日攻擊源於生日問題，問題為當一個房間有 23 個或 23 個以上的人時，那麼至少有 2 個人生日相同的機率要大於一半，這與大多數人的直覺認知相牴觸，故又被稱為生日悖論，但其實他從邏輯角度來看不不是一種悖論。因為大多數人會將房間中 23 個人中任 2 人相同生日的問題，直覺理解為另外 22 個人與你的生日相同的機率，所以才

會產生機率很低的錯覺。而生日攻擊法則是利用 SHA-1 的輸出為 160bit，如果使用暴力破解的話，演算法將需要運行最多 2^{160} 次，生日攻擊法則只需要 2^{80} 次就可以在百分之九十九的機率下找到碰撞。

第 三章　演算法設計

本節通過使用雲端運算系統和 GPU 平行運算系統來描述雜湊值匹配演算法的流程。

在提出的 Hadoop 雲端系統中，接收到主節點分配的數據後，每個工作節點使用 SHA-1 對明文進行雜湊運算，並將其與目標雜湊值進行匹配。類似地，在所提出的 GPU 平行運算系統中，CPU 首先為每個 GPU 執行緒 (thread)分配要處理的數據。因此，在 GPU 系統中，CPU 擔負類似於主節點的作用， GPU 執行緒則類似 Hadoop 系統中的工作節點。

如 Figure 3.1 中簡單雜湊值碰撞演算法的流程圖所示，數字 A 用於為工作節點分配數據範圍為 $A \times range$ 到 $(A + 1) \times range$;range 表示主節點（或 CPU）分配的數據大小; G 是待測文本的序列號; M 是被測試的明文; size 表示明文集的元素的大小，本篇論文是由 26 個小寫的英文字母構成的;Target 表示目標雜湊值; H（M）計算雜湊值，即明文 M 的訊息摘要。

流程圖顯示了每個工作節點（或 GPU 執行緒）上的任務流程。每個節點（或 GPU 執行緒）獲取主節點（或 CPU）分配的數據，然後使用 SHA-1 函數對分配的數據範圍內的明文依序進行雜湊運算，並將每個輸出雜湊值與目標雜湊值進行比較。重複上述循環的雜湊值匹配操作，直到搜索到所分配的整個數據範圍或者目標雜湊值與一些明文的輸出雜湊值相匹配。

在每個節點上執行的雜湊值碰撞演算法主要包括以下步驟：

步驟 1：每個節點接收一個數字 A，表示為 $A \times range$ 到 $(A + 1) \times range - 1$ 的數據分配範圍。

步驟 2：序列號 G 用於產生明文 M. 首先，給 G 分配起始值 $A \times range$。其次，將 G 除以 size 來獲得餘數 r_i，i = 1,2,3,... 和商，並將所獲得的商分配 給 G。重複此步驟直到最新獲得的商等於 0. 第三，通過將剩餘序列 r_n, r_{n-1}... r_3, r_2, r_1 對應到相應的英文字母，即對應 1 到 a，2 到 b，等等。注意，在對應之前，如果任何餘數 $r_k = 0$，則需要用 26（或 size）替換 r_k ，並將下一個餘數 r_{k+1} 減 1，即 $r_k = 26$，$r_{k+1} = r_{k+1}-1$。

步驟 3：執行 SHA-1 演算法獲取雜湊值 H（M）。

步驟 4：將獲得的雜湊值 H（M）與 Target 比較; 如果匹配，傳回匹配的明文 M，否則重複步驟 2 和步驟 3，直到 G = $(A + 1) \times range - 1$。

Figure 3.1 的流程圖所示為每一工作節點的工作，每一節點為雲端系統中的工作節點或 GPU 中的執行緒，每一節點從主節點處取得分配處理的數據，將分配的區間循序進行雜湊並跟標的進行比對，當分配的數據都比對完或找到標的明文，則結束。簡單雜湊值碰撞演算法的基本步驟可以歸納為 Figure 3.2 所示的虛擬碼。

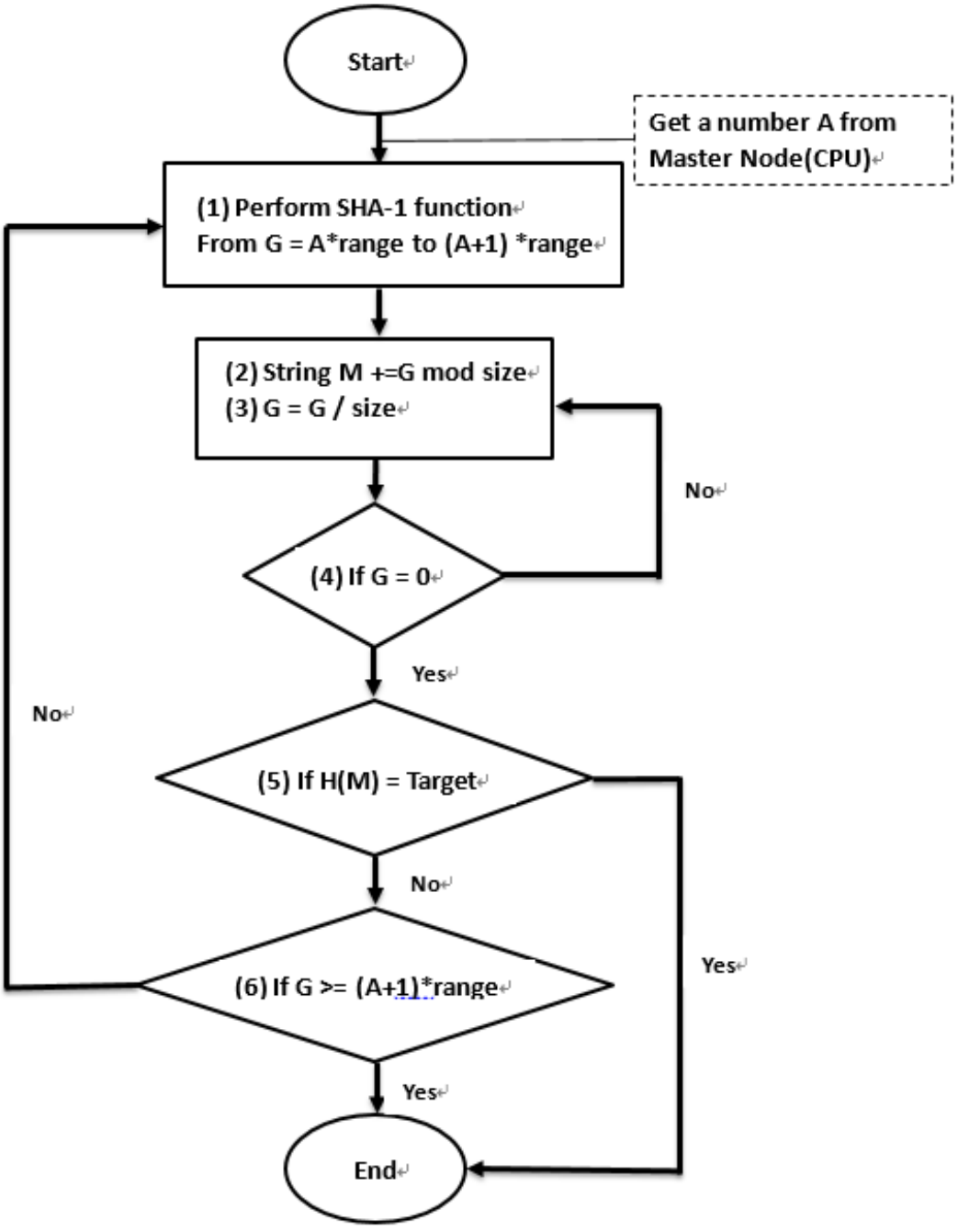


Figure 3.1: 平行運算破解 SHA-1 流程圖

```
Get A from the Master node (or CPU)
matched = 0
do {
for G = A × range to (A + 1) × range − 1
{    do {  M+=G mod size;
           G=G/size }
        } while (G>0)
    Replace number sequence in M with English alphabets to produce plaintext M
    Perform SHA-1 on M to obtain H(M)
    if (H(M)=target)
       {return M
        matched=1 }
} while (matched=0)
```

Figure 3.2: 單一節點（或執行緒）上的簡單雜湊值碰撞演算法的虛擬碼

第 四章　實驗結果

在本研究中，我們設計各項實驗分析了雲端系統和 GPU 的運行時間。通過實驗得出的效率與所使用的資源進而推知找出 SHA-1 的碰撞所需要的資源與時間關係。

4.1 實驗環境

Hadoop 平台由一個主節點組成，具有 8G RAM 的 G860 CPU 的實體電腦和三個子節點，每個子節點都使用帶有 8G RAM 的 i7-4790 CPU 的實體電腦。每個子節點具有 3 個 VM，1 個 CPU 和 2G RAM，所有節點均使用 ubuntu14.04。使用三台實體電腦，因此最多可以有 9 個節點。

GPU 的實驗環境為 NVIDIA GeForce GTX650Ti 在具有 20G RAM 的 Intel(R) Core(TM) i5-4690 CPU 的 Windows 10 作業系統，NVIDIA GeForce GTX650Ti 擁有 576 個核心，1GB 的 GDDR5 記憶體。如 Table 4.1。

Table 4.1: 實驗環境

	CPU	GPU	RAM	作業系統	備註
雲端主節點	G860		8G	ubuntu14.04	
雲端子節點	i7-4790		8G	ubuntu14.04	共 3 台，每台 3 個 VM
雲端子節點 VM	i7-4790		2G	ubuntu14.04	每台 VM 均為單核心
GPU 環境	i5-4690	GTX650Ti	20G	Windows 10	

4.2 實驗

4.2.1 雲端環境中單一節點不同數據量下的運行速度的研究

首先研究單一節點在不同數據量下的表現，雲端環境下，主節點分配的資料傳輸次數越多，導致花費在傳輸的時間越多，使雲端工作效率越低，故每次分配的數據量要越多越好。而為了即時取得破解的明文，減少多餘的工作，需要增加檢查的頻率，所以每次分配的資料量盡量越少越好。所以實驗的目標是在效率不降低的情況下，每次分配的數據量越少越好。

根據 Table 4.2, Table 4.3, Table 4.4, Table 4.5 所示，雲端的單一節點工作效率在 50 萬到 70 萬之間，如 Figure 4.1 當資料吞吐量小於 1000 萬時，傳輸時間佔運行時間的比重會大幅上升，使工作效率降低，當單一節點每次執行的次數在 4000 萬到 5000 萬最符合研究需求，即單一節點每秒可以執行 60 萬到 65 萬次。

Table 4.2: 雲端環境中單一節點不同數據量下的運行時間 (十億到一百億次) (B: 十億)

	1B	2B	3B	4B	5B	6B	7B	8B	9B	10B
時間 (秒)	1566	2969	4175	6325	7986	8666	9180	10632	12432	13996
速率 (萬次/秒)	63.86	67.36	71.86	63.24	62.61	69.24	76.25	75.24	72.39	71.45

Table 4.3: 雲端環境中單一節點不同數據量下的運行時間 (一億到十億次) (HM: 一億)

	1HM	2HM	3HM	4HM	5HM	6HM	7HM	8HM	9HM	10HM
時間 (秒)	144	326	465	555	808	948	1065	1257	1385	1355
速率 (萬次/秒)	69.44	61.35	64.52	72.07	61.88	63.29	65.73	63.64	64.98	73.80

Table 4.4: 雲端環境中單一節點不同數據量下的運行時間 (一千萬到一億次) (TM: 千萬)

	1TM	2TM	3TM	4TM	5TM	6TM	7TM	8TM	9TM	10TM
時間 (秒)	17	34	53	62	80	100	99	125	146	140
速率 (萬次/秒)	58.82	58.82	56.60	64.52	62.50	60.00	70.71	64.00	61.64	71.43

Table 4.5: 雲端環境中單一節點不同數據量下的運行時間 (一百萬到一千萬次)(M: 百萬)

	1M	2M	3M	4M	5M	6M	7M	8M	9M	10M
時間 (秒)	6	6	8	10	11	15	14	18	17	18
速率 (萬次/秒)	16.67	33.33	37.50	40.00	45.45	40.00	50.00	44.44	52.94	55.56

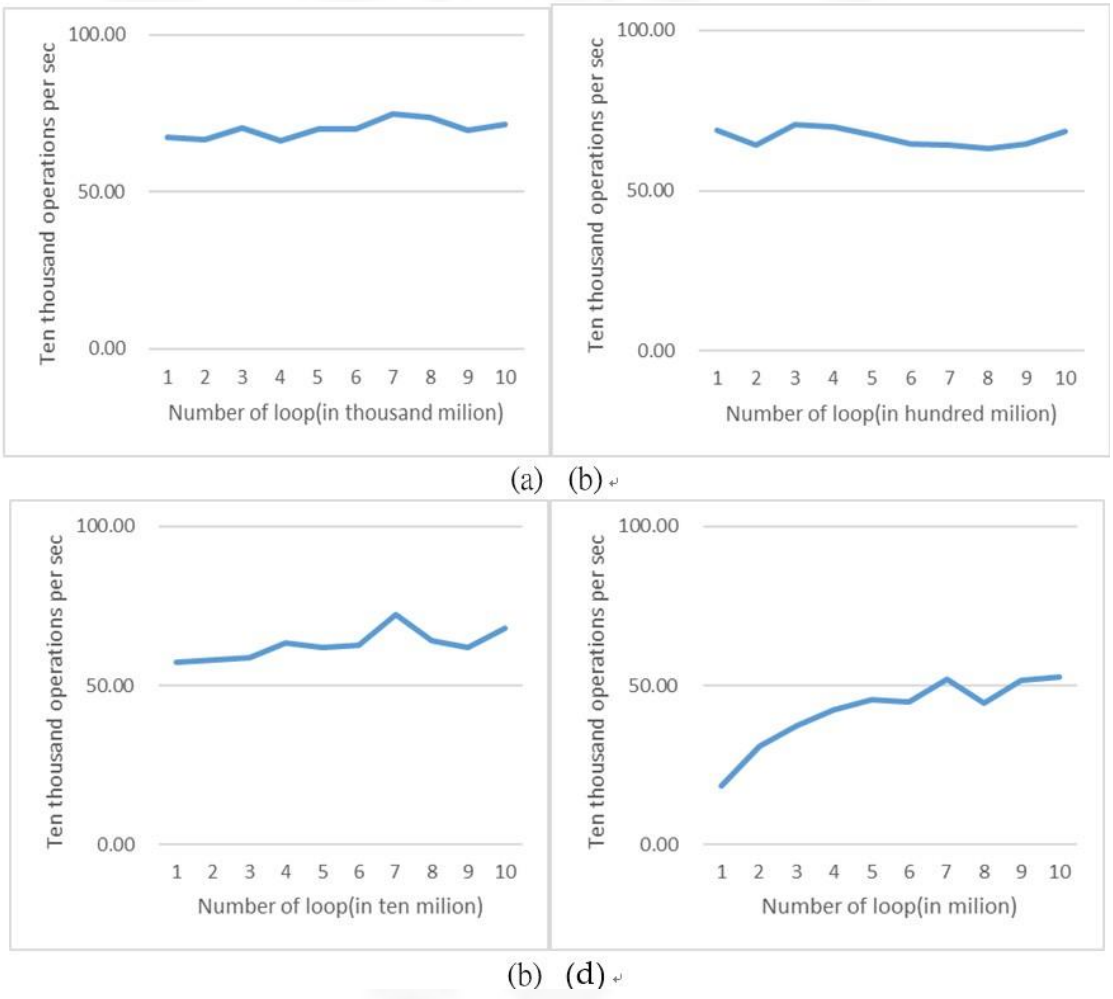


Figure 4.1: 雲端環境單一節點每次分配不同數據量的運行速度研究，它們分別是: (a) 十億到百億次 (b) 一億到十億次 (c) 一千萬到一億次 (d) 一百萬到一千萬次

4.2.2 雲端環境下不同節點數的效能比較

在實驗 2 中，在不同的節點數下，研究系統的破解效能，如 Figure 4.2 所示，隨著節點增加，效能逐漸增加。再看 Table 4.6，每增加一個節點，速率比平均增加 0.7 到 0.87 之間，且隨著資料吞吐量及節點的增加，平均每台的節點速率比會再增加，故在雲端環境下，每個節點可以發揮 0.8 台電腦的效能，且

隨著資料吞吐量或節點的增加，最大可達 0.87 台電腦的效能。Figure 4.3 則顯示 9 個節點數在不同資料量下與單一節點的加速比，公式如 Equation 4.1。

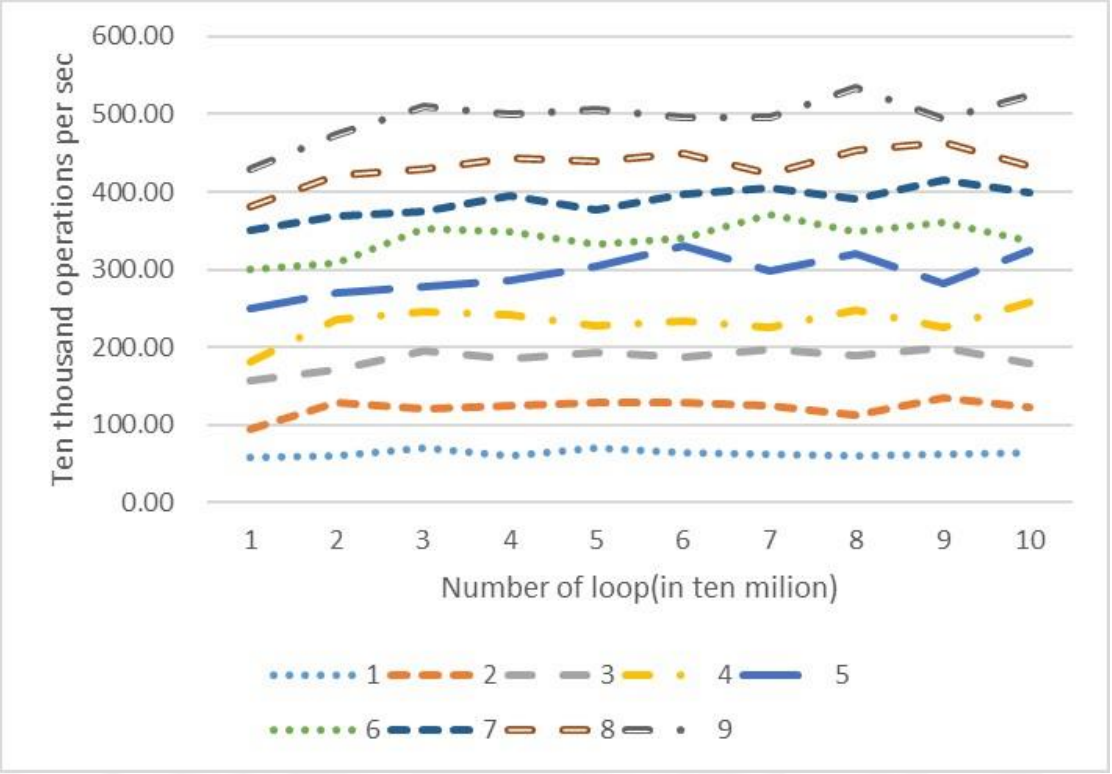


Figure 4.2: 雲端環境不同節點數的效能比較

$$\text{雲端加速比}(Speedup) = \frac{\text{1 VM 執行所需的時間}}{\text{k VMs 執行所需的時間}} \tag{4.1}$$

Table 4.6: 雲端環境不同節點數的加速比 (TM: 千萬)

	1TM	2TM	3TM	4TM	5TM	6TM	7TM	8TM	9TM	10TM
1 node	0.88	0.91	1.05	0.91	1.04	0.96	0.92	0.91	0.95	0.95
2 nodes	1.43	1.94	1.80	1.88	1.93	1.94	1.88	1.71	2.02	1.86
3 nodes	2.37	2.58	2.94	2.78	2.89	2.82	2.98	2.84	3.01	2.69
4 nodes	2.73	3.54	3.68	3.65	3.42	3.50	3.40	3.73	3.40	3.88
5 nodes	3.76	4.06	4.18	4.30	4.58	4.96	4.50	4.81	4.26	4.88
6 nodes	4.51	4.63	5.31	5.23	5.01	5.11	5.59	5.23	5.41	5.07
7 nodes	5.26	5.54	5.64	5.93	5.66	5.96	6.09	5.89	6.23	6.02
8 nodes	5.73	6.33	6.44	6.68	6.61	6.75	6.38	6.83	6.99	6.50
9 nodes	6.44	7.12	7.66	7.52	7.60	7.45	7.46	8.02	7.43	7.87

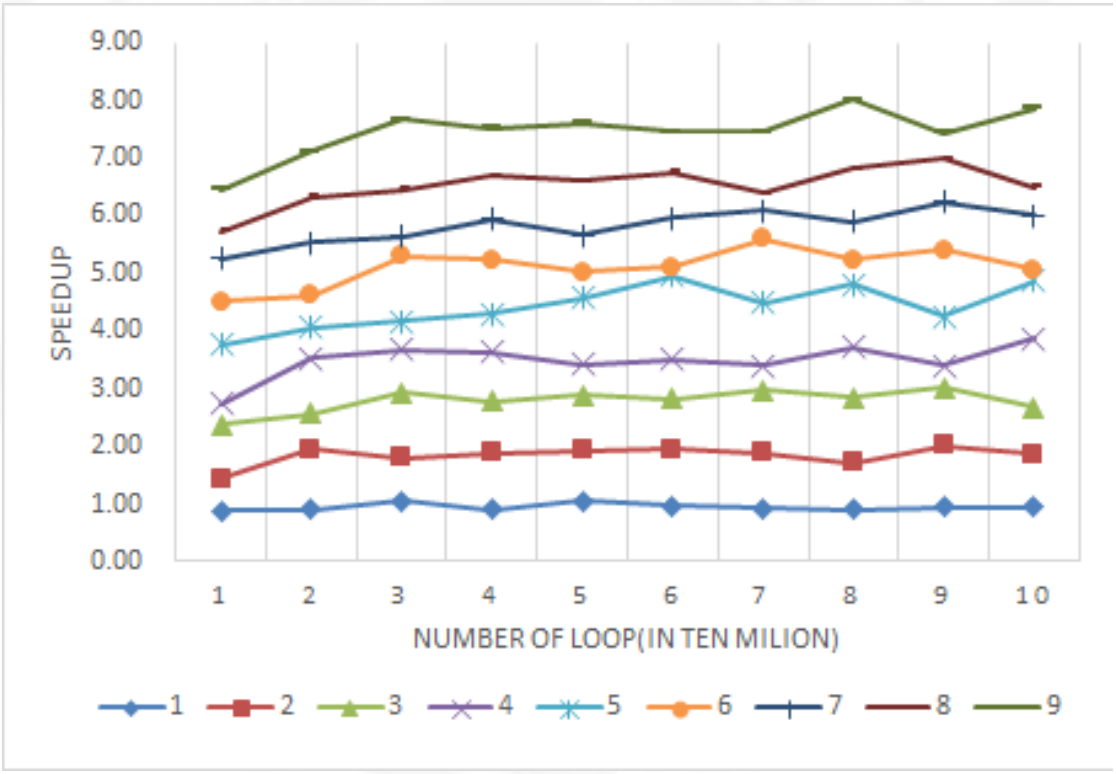


Figure 4.3: 雲端環境不同節點數的加速比

4.2.3 雲端環境下不同節點數破解不同字元數所需的時間研究

使用不同數量的節點來研究系統破解不同字元數的執行時間，Figure 4.4 顯示了用於破解字元長度為 6 個字元的明文在各個節點數的執行時間，因為在 IoT 中，所使用的密碼長度通常不會太長，通常為 6 個字元到 10 個字元，故實驗測試了 6 個字元密碼長度由此得到在 IoT 使用 SHA-1 雜湊密碼的安全性。執行時間隨著節點數量的增加而減少。Figure 4.5 顯示了用於破解不同字元數的執行時間。

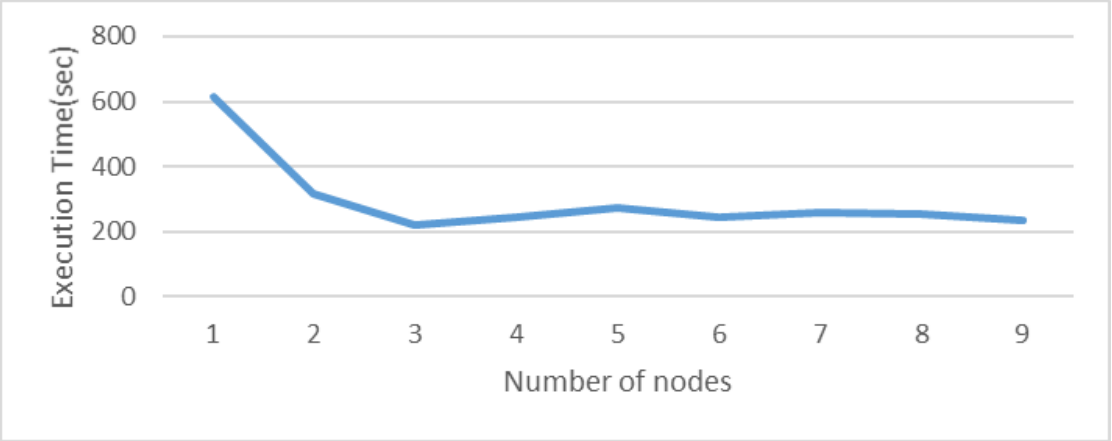


Figure 4.4: 雲端環境下不同節點數破解 6 個字元數所需的時間

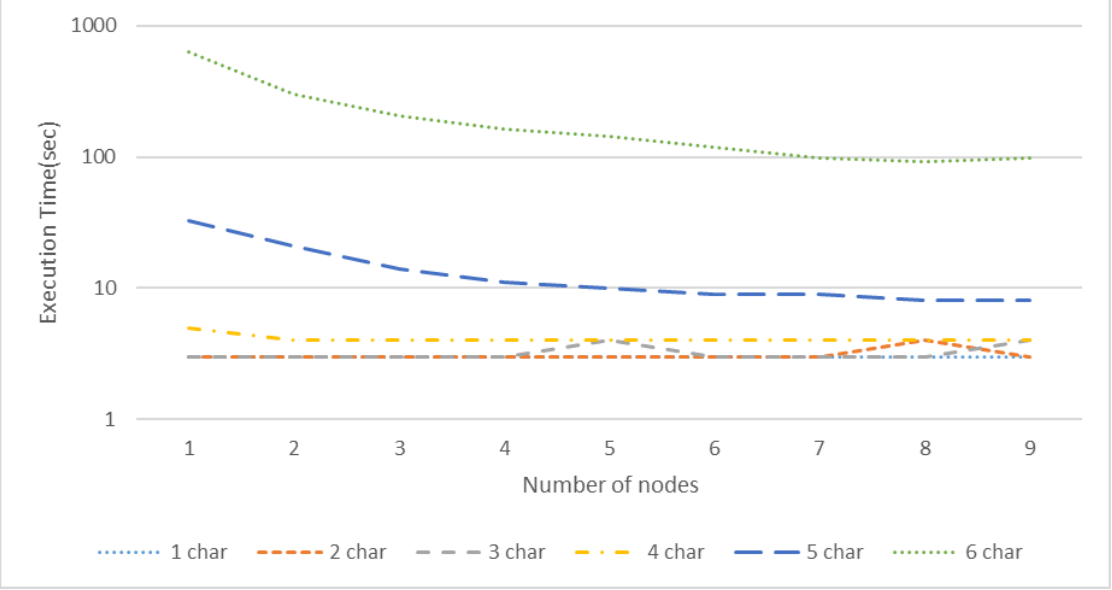
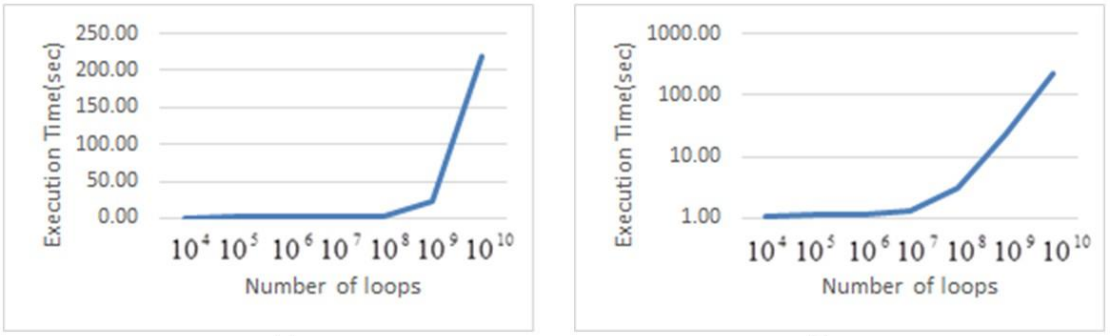


Figure 4.5: 雲端環境下不同節點數破解不同字元數所需的時間

4.2.4 GPU 相同執行緒不同數據量執行時間研究

首先研究相同執行緒的不同執行次數的執行時間，Figure 4.6(a) 表示為 1024 個執行緒分別在 10 的 4 次方到 10 次方的執行次數所需要的執行時間，Figure 4.6(b) 則是將 Figure 4.6(a) 的時間刻度以 10 的冪次方表示，發現當 1024 個執行緒在執行次數低於 10 的 7 次方時，所花的時間幾乎一樣，而當執行次數超過 10 的 8 次方時，執行時間跟執行次數成正比。



(a) (b)

Figure 4.6: GPU 1024 執行緒破解 10 的 4 次到 10 的 10 次所需時間，(b) 將時間取對數後的折線圖

4.2.5 GPU 不同執行緒數量的效能比較

在這個實驗中，在不同的執行緒數下，研究系統的破解效能，如 Figure 4.7 所示，隨著執行緒的增加，效能提高。當執行緒超過 4096 個時，速度沒有增加，顯示 GTX650 雖然可以支援到 8192 個執行緒，但是因為共享記憶體的關係，大於 4096 個執行緒之後就沒有顯著的成長。

Table 4.7 為當 GPU 使用不同數量的執行緒執行 10^9 次所需要的時間。

Table 4.7: GPU 不同執行緒數量的執行時間

	1024	2048	4096	8192
時間	22.51	11.79	6.50	6.54

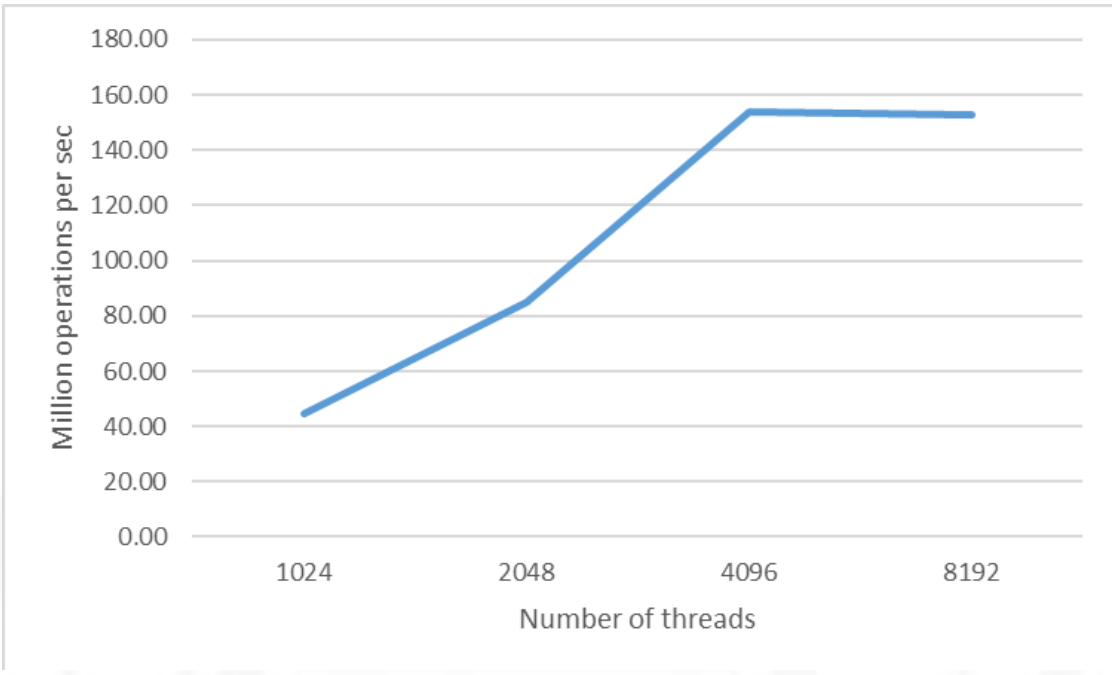


Figure 4.7: GPU 不同數量的執行緒間的效能折線圖

4.2.6 GPU 不同執行緒數量與破解不同字元數的執行時間研究

使用不同數量的執行緒來研究系統破解不同字元數的執行時間，Figure 4.8 顯示了用於破解字元長度為 7 個字元的明文在各個執行緒數的執行時間。執行時間隨著執行緒的增加而減少。

Table 4.8 為當 GPU 使用不同數量的執行緒用於破解 7 個字元所需要的時間。

Table 4.8: GPU 不同執行緒數量破解不同字元數的執行時間

	1	2	3	4	5	6	7
1024	0.81	1.68	2.23	2.95	3.90	11.32	182.42
2048	0.81	1.67	2.24	2.91	3.86	7.81	94.09
4096	0.83	1.67	2.27	2.98	3.89	6.03	49.49
8192	0.82	1.71	1.99	2.99	3.93	6.35	50.39

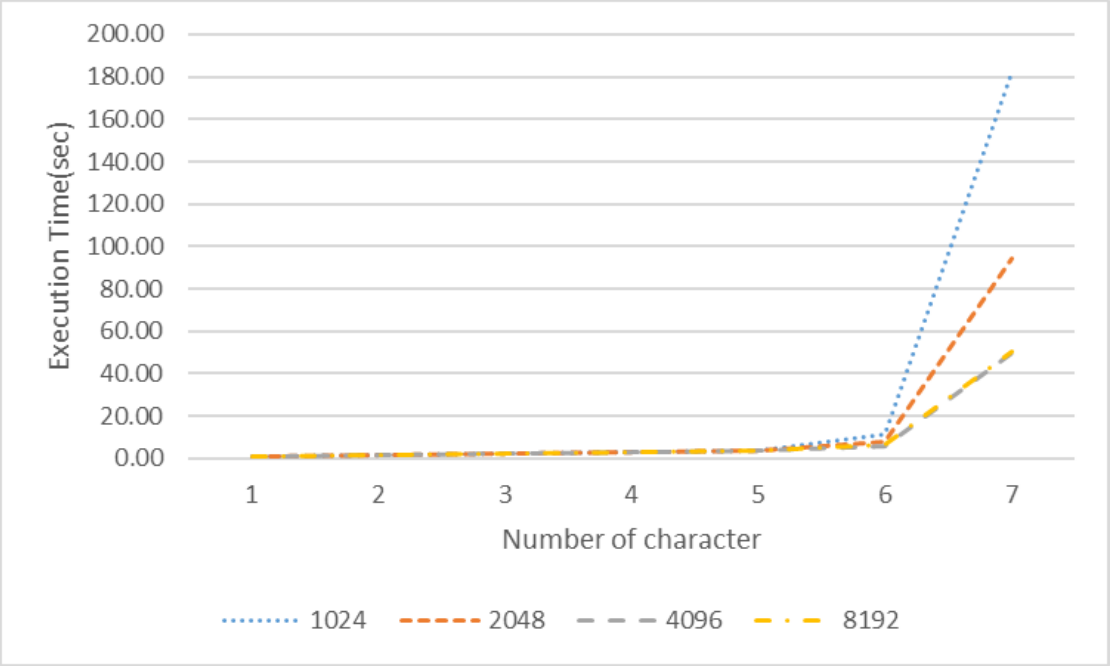


Figure 4.8: GPU 不同數量的執行緒破解不同字元數的時間折線圖

4.2.7 雲端系統與 GPU 平行運算的加速比

如 Table 4.9 所示，研究單機、雲端不同節點數、GPU 不同執行緒在破解 7 個字元的執行時間，並算出與單機的速率比。當 GPU 使用 4096 個執行緒時，其速度為單機不使用 GPU 還快 103 倍，相當於 344 台 VM 組成的 Hadoop 雲端系統。Figure 4.9 為單機與雲端的加速比，可以看到 3 個節點與單機的加速比差異不大，這是因為一台實體電腦使用了 3 個 VM 當作節點，Figure 4.10 則是單機與 GPU 的加速比，可以看出 GPU 對於雜湊的運算，其運算效率明顯大於單機 20 到 100 倍。

$$\textit{加速比(Speedup)} = \frac{\text{對照組 (單機) 執行所需的時間}}{\text{實驗組執行所需的時間}} \tag{4.2}$$

Table 4.9: 雲端環境、GPU 與單機的加速比

	時間 (秒)	加速比
單機	5130	1.00
3 nodes	4836	1.06
6 nodes	2727	1.88
9 nodes	1890	2.71
1024 threads	182	28.12
2048 threads	94	54.52
4096 threads	49	103.66

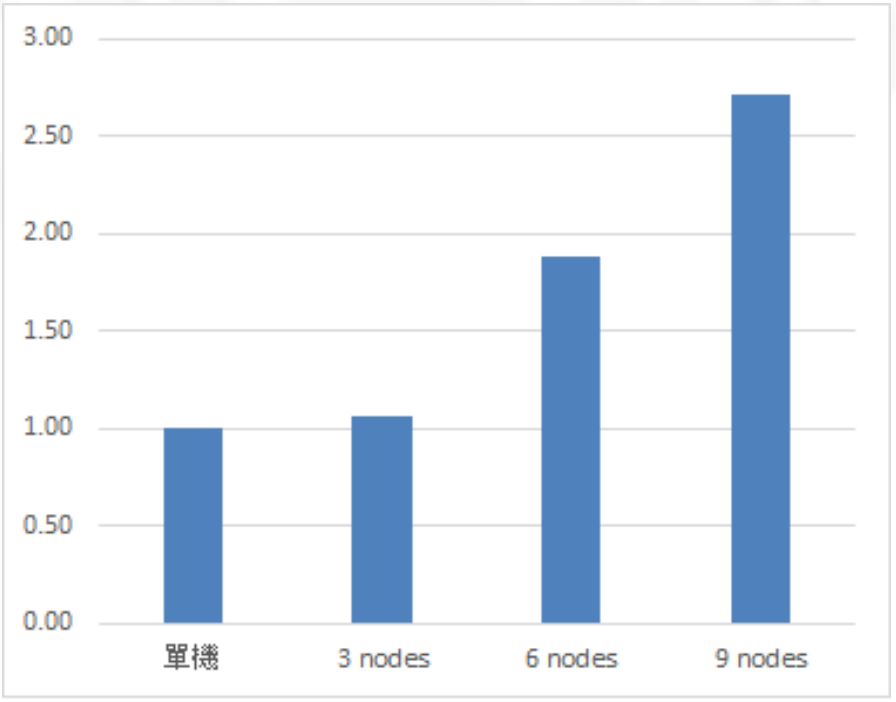


Figure 4.9: 單機與雲端的加速比

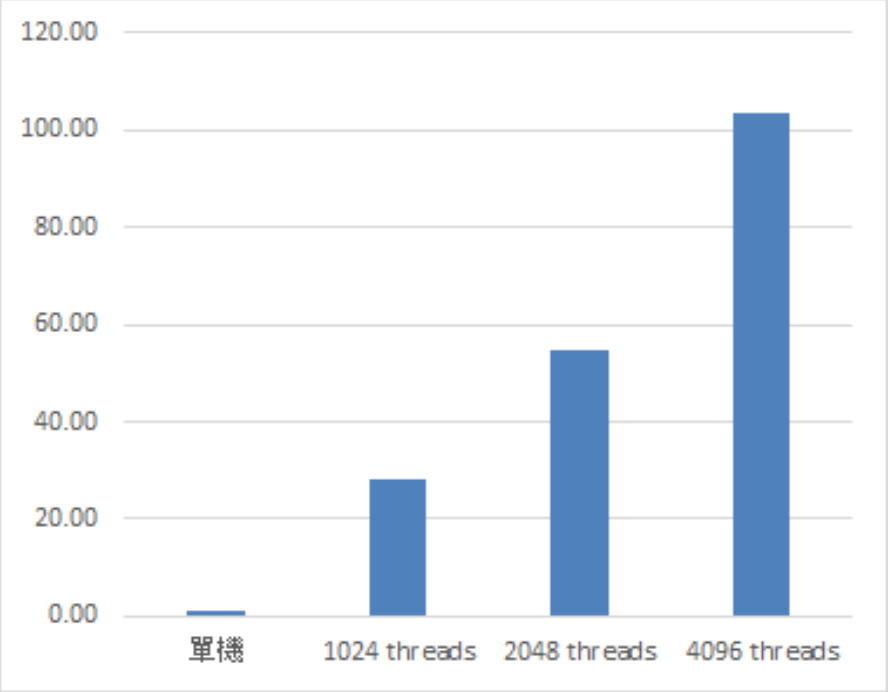


Figure 4.10: 單機與 GPU 的加速比

第 五章　結論

IoT 佈署通常使用雜湊函數進行安全認證，為了瞭解其中的安全威脅，研究了執行簡單雜湊值運算的演算法在雲端環境跟 GPU 平行運算的效能。根據實驗結果，在雲端環境中，每個節點每秒可以比對 60 萬到 65 萬次，由於在 IoT 中所使用的密文長度大約在 6 到 10 個字元左右，所以，如果使用暴力破解法來尋找 SHA-1 10 個字元以內的碰撞，使用 9 個 VM 的話，最多需要 279 天找到 SHA-1 的碰撞，而若使用 2511 個 VM(837 台實體機) 就能在 1 天內找到，如果使用更多的節點、更高性能的電腦、演算法和破解方法，破解的時間會更加縮短。如今，諸如 Google 這樣的雲端服務提供商擁有超過一百萬個節點，而雲端節點使用的成本較低，且佈署速度更快，從而使對手更容易佈署基於雲端的破解系統。在 GPU 平行運算中，每個執行緒每秒可以比對 43000 次到 44000 次，一張 GTX650 Ti 可運行 8192 個執行緒但是因為共享記憶體限制，大約只有 4096 個執行緒，這大約跟擁有 344 個節點的雲端系統相當，雖然佈署成本較低，但是佈署不易。如能結合雲端系統與 GPU 平行運算，使每一節點均使用 GPU 運算，便能使破解速度加快，總而言之，SHA-1 的安全性不再可靠。

參考文獻

1. H. Khemissa, D. Tandjaoui, “A Novel Lightweight Authentication Scheme for Heterogeneous Wireless Sensor Networks in the Context of Internet of Things,” Proceedings of 2016 Wireless Telecommunications Symposium, London, United Kingdom, April 18-20, 2016.
2. C. H. Lin, W. S. Hsieh, F. Mo, M. H. Chang, “A PTC Scheme for Internet of Things: Private-Trust-Confidentiality,” Proceedings of the 30th International Conference on Advanced Information Networking and Applications Workshops, Crans-Montana, Switzerland, March 23-25, 2016.
3. Xiaoyun Wang, Dengguo Feng, Xuejia Lai, Hongbo Yu. “Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD,” Cryptology ePrint Archive: Report 2004/199 2004, Available online: http://eprint.iacr.org/2004/199 (accessed on 24 12 2016).
4. R. K. Ibraheem, R. A. J. Kadhim, A. S. H. Alkhalid. “Anti-collision Enhancement of a SHA-1 Digest using AES Encryption by LABVIEW,” Proceedings of 2015 World Congress on Information Technology and Computer Applications, Hammamet, Tunisia, 11-13 June 2015.
5. M. Hassan, A. Khalid, A. Chattopadhyay, C. Rechberger, T. Güneysu, C. Paar. “New ASIC/FPGA Cost Estimates for SHA-1 Collisions,” Proceedings of the 18th Euromicro Conference on Digital Systems Design, Madeira, Portugal, August 26-28, 2015.
6. J. Fuß, S. Gradinger, B. Greslehner-Nimmervoll, R. Kolmhofer. “Complexity Estimates of a SHA-1 Near-Collision Attack for GPU and FPGA,” Proceedings of the 10th International Conference on Availability, Reliability and Security, Toulouse, France, August 24-27, 2015.
7. Yusuf Moosa Motara, B. Irwin. “SHA-1 and the Strict Avalanche Criterion,” 2016 Information Security for South Africa (ISSA), Johannesburg, 2016, pp. 35-40.
8. I. Muttik, C. Barton, “Cloud Security Technologies,” Information Security Technical Report 2009, Volume 14, Issue 1, pp. 1-6.
9. L. M. Kaufman. “Data Security in the World of Cloud Computing,” IEEE Security & Privacy 2009, Volume 7, Issue 4, pp. 61-64.
10. B. R. Kandukuri, R. P. V., A. Rakshit. “Cloud Security Issues,” Proceedings of the 2009 IEEE International Conference on Services Computing, Washington DC, USA, September 21-25, 2009.
11. Arne Jørgen Berre, Dumitru Roman, Einar Landre, Willem-Jan van den Heuvel, Lars Arne Skår, Morten Udnæs, Ruth Lennon, Amir Zeid. “Towards Best Practices in Designing for the Cloud,” Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications, Florida, USA, October 25-29, 2009.
12. Yifeng Luo, Siqiang Luo, Jihong Guan, Shuigeng Zhou. “A RAMCloud Storage System based on HDFS: Architecture, Implementation and Evaluation,” Journal of Systems and Software, 2013, Volume 86, Issue 3, pp. 744–750.
13. Apache Hadoop. Available online: http://hadoop.apache.org/ (accessed on 24 12 2016).

14. Dongyu Feng, Ligu Zhu, Lei Zhang. “Review of Hadoop Performance Optimization,” 2016 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, 2016, pp. 65-68.
15. R. R. Parmar, S. Roy, D. Bhattacharyya, S. K. Bandyopadhyay, T. H. Kim. “Large Scale Encryption in Hadoop Environment: Challenges and Solutions,” in IEEE Access , vol.PP, no.99, pp.1-1
16. T. D. Han, T. S. Abdelrahman. “hiCUDA: High-level GPGPU Programming,” IEEE Transactions on Parallel and Distributed Systems, Volume 22.1, pp. 78-90, 2011.
17. R. Ubal, B. Jang, P. Mistry, D. Schaa, D. Kaeli. “Multi2Sim: A Simulation Framework for CPU-GPU Computing,” 2012 21st International Conference on Parallel Architectures and Compilation Techniques (PACT), Minneapolis, MN, 2012, pp. 335-344.
18. A. Cano, C. García-Martínez. “100 Million Dimensions Large-Scale Global Optimization using Distributed GPU Computing,” 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, 2016, pp. 3566-3573.
19. E. Lindholm, J. Nickolls, S. Oberman, J. Montrym. “NVIDIA Tesla: A Unified Graphics and Computing Architecture,” IEEE Micro, Volume 28, pp. 39-55, 2008.
20. Cheng-Chieh Li, Jung-Chun Liu, Chu-Hsing Lin, Winston Lo. “On the Accelerated Convergence of Genetic Algorithm Using GPU Parallel Operations,” International Journal of Software Innovation Volume 3(4), pp. 1-17, 2015Applications”, May 26, 2016