分散式雲端期中作業

R10725012 呂晟維

Data Migration (33%)

- Functionality

這項功能是在新節點加入時，剛初始化的新節點 n 的 Filesystem 會是空的。但因為 n 的 success 原本負責範圍是了 predeccesor_id ~ success_id 的 key range，在新節點 n 加入後，n 需要負責的 key range 範圍是 predeccesor_id ~ n_id，而 n 的 success 把負責範圍縮小至 n_id+1 ~ success_id。這個過程稱為 Data Migration。

- Component

為了達成 Data Migration 我製作一個 do_migrate.py 負責維護這個節點資料的搬出功能，因此不需要再大費周章實作搬入功能，do_migrate.py 與 init_script.py 一樣位於使用者的資料夾目錄頂端。定期檢查的具體方法是 do_migrate.py 會每分鐘檢查一次 /files 底下的所有檔案，查看是否有檔案需要搬動，若 (hash(filename) -> find_successor() -> succ_ip) 得到的 succ_ip 跟自己的 ip 不同，則表示有其他節點負責了這個 file key，需要 migrate 給對方。如果需要搬動則先確認 succ_ip 沒有這個檔案後，上傳該檔案至 succ_ip，刪除自己本地的檔案。為了避免請求同時發出，每上傳一個檔案會隨機 sleep 幾秒。

ps: 若要增加 replica 功能，持有備份的機器就不能在 migrate 時進行刪除動作，詳細見第三題。

```
# Data Migration 使用的元件
ec2-user/
|-- ini_script.py
|-- do_migrate.py   # Data Migration 功能模組在這
|-- do_localfile_checkserver.py  # 提供 filecheck GET API
|-- chord
|-- files/
    |-- ...         # 這個 instance 的 Filesystem 中的檔案放這
```

- Experiment

創建3個節點以上的 Ring 稱 A, B, C 三台ec2機器，連接方向是 A→B→C→。我們準備一個 a.txt 檔案對他進行 find_successor()，假如得到了 A 的ip 表示 A 負責這個 filekey。我們故意「上傳錯機器」給 B a.txt，當 B 的 do_migrate.py 例行檢查到 a.txt 其實是 A 要負責的，則會把 a.txt 上傳給 A。

File Chunks (33%)

- Functionality

ps: 這個流程是在地端執行的，地端是指上傳者的電腦。(當然也可以在ec2上執行，此時ec2就視為地端)

這項功能是實現節點之間的 Load balancing，若欲上傳的檔案大小大於 4 MB (file.txt)，則先在本地被切成多個 4 MB 以及1個 0~4 MB 的剩餘檔案，將新增的切片檔案命名為 file.txt-part1, file.txt-part2, … , file.txt-partN 放置在 /chunks，最後依據各檔案 filename 來 find_successor() 上傳到指定的節點。(若檔案小於 4 MB 則只改命名為 file.txt-part1，不做切片)

下載檔案時同理，若欲下載的檔名為 file.txt，則先去負責的節點尋找 file.txt-part1 (也尋找 file.txt 防呆)，若有找到第一份檔案表示該檔案存在切片並被上傳過，再依序尋找 file.txt-part2, file.txt-part3 … 直到檔案找不到後續的切片，那麼就可以斷定切片有 1 ~ N 份。下載完數個切片檔至 /chunks 資料夾再組裝起來得到 file.txt。

- Component

如架構圖，為了本功能多寫的檔案為 upload_chunk.py, download_chunk.py，一個負責將檔案切片呼叫 upload.py 執行上傳，另一個負責檢查雲端分散式系統上切片檔呼叫 download.py 執行下載。

```
# File Chunks 使用的元件
ec2-user/
|-- ini_script.py
|-- do_localfile_checkserver.py  # 提供 filecheck GET API
|-- chord
|-- chord-part-2/
    |-- upload.py           # 上傳單一檔案 (by助教)
    |-- upload_chunk.py     # 切片大檔案，並呼叫 upload.py 執行上傳
    |-- download.py         # 下載單一檔案 (by助教)
    |-- download_chunk.py   # 尋找切片過的檔案，並呼叫 download.py 執行下載
    |-- chunks/
        |-- ...             # 暫存切片檔的資料夾
|-- files/
    |-- file.txt-part1      # 這個 instance 的 Filesystem 中的檔案放這
```

- Experiment

創建3個節點以上的 Ring 稱 A, B, C 三台ec2機器，初始時檔案系統都是空的。

新建一個 6MB 大小的 file.txt，執行 upload_chunk.py 做上傳的動作，程式會 print 出 file.txt-part1 和 file.txt-part2 被上傳到哪裡。再刪除本地的 file.txt 執行 download_chunk.py 會看到檔案順利下載回來。

Replication (33%)

- Functionality

這項功能是在讓每一個檔案都有多個備份分別存放在 ring 上不同的 ec2 instance 中 (不論有沒有切file chunks)。預設的備份數量為 3 份，也就是說當使用者上傳 1 個檔案 f，整個分散式系統最終會儲存 3 份檔案 f 分別在 3 台機器上，這個備份數量是可以調整的，只要簡單修改 do_replica.py 中的變數即可。我的思路是只讓負責檔案 f 的第一個節點 (也就是successor of file id)來監控檔案 f，其餘兩個備份節點只需要接收備份的 upload() 請求就好。

- Component

為了達成 Data replica 我製作一個 do_replica.py 負責維護這個節點資料(primary file)的備份功能，primary file 是指當我們對 Filesystem 底下的一個檔案名稱執行 find_successor() 時，回傳的節點是自己，那麼這個檔案就是這個節點的 primary file (如架構圖的 a.txt)，當然這個節點也會儲存來自其他節點的 replica (如架構圖的 back.txt)。

每個節點會定時(預設1分鐘)對 /files 底下的所有 primary file 檔案檢查他額外2個備份是否有存在。具體方式是呼叫 get_successor(0), get_successor(1) 來取得 Ring 中直連的下兩個節點為備份節點，檢查他們的 Filesystem 是否有這個 primary file；do_localfile_checkserver.py 是個簡易的 API 伺服器提供了一個布林值的 get /filename?=xxx API，通過呼叫這個 API，能簡單的判斷備份是否存在，而不用真的去下載檔案。

針對非 primary file 檔案 (即 replica 檔案)，每個節點也可以透過 get API 檢查主要節點是否有這個檔案，沒有也還原一份上去。其實這個就是第一題 Migration 的功能，只要讓 migrate 時不要刪除自己本機的 replica 檔案就好，若要防止 redundant 的備份(如機器D備份了第四份)，可以建一個 list 給 do_localfile_checkserver.py 若檔案一段時間(如10分鐘)沒有被別人檢查到則刪除該檔案 。

```
# Data Replication 使用的元件
ec2-user/
|-- ini_script.py
|-- do_replica.py   # Data Replication 功能模組在這
|-- do_localfile_checkserver.py  # 提供 filecheck GET API
|-- chord
|-- files/
    |-- a.txt       # assume primary file
    |-- back.txt    # assume replica from other ec2
    |-- ...
```

- Experiment

創建3個節點以上的 Ring 稱 A, B, C 三台ec2機器，初始時檔案系統都是空的。透過 find_successor() of a.txt 先在 A 機器上傳 a.txt。等待至少一分鐘讓 replica 機制完成執行，去 B 機器和 C 機器檢查備份。

此時可以刪除 A 的 a.txt 並停掉 chord，再等待至少兩分鐘重啟 chord 我們會看到 A 機器上的 a.txt 被還原回來了。(因為刪除時 a.txt 會變成 B 機器的 primary file 交由 B 檢查備份)

附錄

▼ 前置動作

```
aws configure set aws_access_key_id ASIA5MU4OC5PDWOGNDUQ
aws configure set aws_secret_access_key HsT...
aws configure set aws_session_token ...
```

▼ Auto-scale demo

1. 監控群組的磁碟空間 `xvda1`，先設一個 Group > Simple Group Policy 讓機器加一，再設 CW > alarm 指定磁碟使用率 `33%`。

- 約 62MB 的檔案 Irish CoNLL17 corpus，解壓縮完產生 80MB 的 model.txt。

```
wget http://vectors.nlpl.eu/repository/20/51.zip  # 62MB
unzip 51.zip                                      # disk_used_percent: 32.2% origin, 33.5% zip, 3
3.95% after upload
ls -l --block-size=M
python3 /home/ec2-user/chord-part-2/upload.py model.txt 172.31.41.118  # 上傳讓自己的chord指派位置

wget https://pjreddie.com/media/files/yolov3.weights (237MB)
```

▼ How to register crontab to schedule migrate & replica?

- 先把預設文字編輯器改掉 `export EDITOR=nano`
- `crontab -e` 開啟vim修改，輸入 command，vim :wq
- https://phoenixnap.com/kb/how-to-vim-save-quit-exit
- `crontab -l` 確認排程工作，做成 image後，之後每台機器上都有 migrate & replica 功能了。

```
[root@ip-172-31-39-65 ~]# crontab -l
* * * * * /usr/bin/python3 /home/ec2-user/do_migrate.py >> /home/ec2-user/log_migrate.txt 2>&1
* * * * * /usr/bin/python3 /home/ec2-user/do_replica.py >> /home/ec2-user/log_replica.txt 2>&1
```

▼ Endpoints of API 伺服器 (do_localfile_checkserver.py)

- `http://3.91.238.43:9999/list_files` GET
- `http://127.0.0.1:9999/file_exists?filename=aaa.txt` GET

```
// Respond of http://3.91.238.43:9999/list_files

{
  "files": [
    "tmp.txt"
  ]
}
// Respond of http://127.0.0.1:9999/file_exists?filename=aaa.txt
{
  "file_exists": true
}
```

▼ Migrate demo

- 兩台機器 p3a, p3b `cd /home/ec2-user` `ls files`
- 先對 “a.txt” find_successor() 發現是 p3b 機器負責。

```
[root@ip-172-31-34-248 ec2-user]# python3 act_findsucc.py 172.31.34.248 a.txt
info[0]: b'172.31.34.248' type: <class 'bytes'>
find_successor id: 607204990's successor is 172.31.34.248
well done

[root@ip-172-31-34-248 ec2-user]#
```

i-0437a0cb7bd73b8ea (p3b)

PublicIPs: 3.91.73.206 PrivateIPs: 172.31.34.248

- 所以故意傳給 p3a “a.txt”

```
[root@ip-172-31-34-1 ec2-user]# touch files/a.txt
[root@ip-172-31-34-1 ec2-user]# ls files
a.txt
[root@ip-172-31-34-1 ec2-user]#
```

i-01246383ec50c3500 (p3a)

PublicIPs: 54.82.195.77 PrivateIPs: 172.31.34.1

- 等待1分鐘去檢查 p3b，發現 p3b 被 migrate 機制補上了 “a.txt”。

```
[root@ip-172-31-34-248 ec2-user]# ls files
[root@ip-172-31-34-248 ec2-user]# ls files
a.txt
[root@ip-172-31-34-248 ec2-user]#
```

i-0437a0cb7bd73b8ea (p3b)

PublicIPs: 3.91.73.206 PrivateIPs: 172.31.34.248

```
# 常用指令
python3 init_script.py
python3 do_localfile_checkserver.py
python3 act_with_ec2.py 節點ip
python3 do_migrate.py
rm -rf files/*
touch files/tmp.txt
touch files/at3a.txt
```

▼ Replica demo

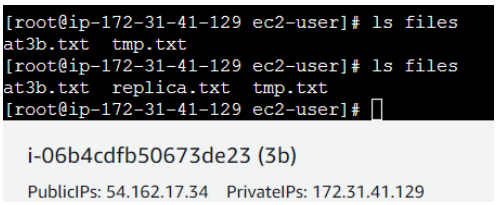
1. 測試思路跟 migrate 相反，先對 “replica.txt” find_successor() 發現是 p3a 機器負責。

```
[root@ip-172-31-41-129 ec2-user]# python3 act_findsucc.py 172.31.41.129 replica.txt
info[0]: b'172.31.41.129' type: <class 'bytes'>
find_successor id: 1130095303's successor is 172.31.39.176
well done
```

i-06b4cdfb50673de23 (3b)

PublicIPs: 54.162.17.34 PrivateIPs: 172.31.41.129

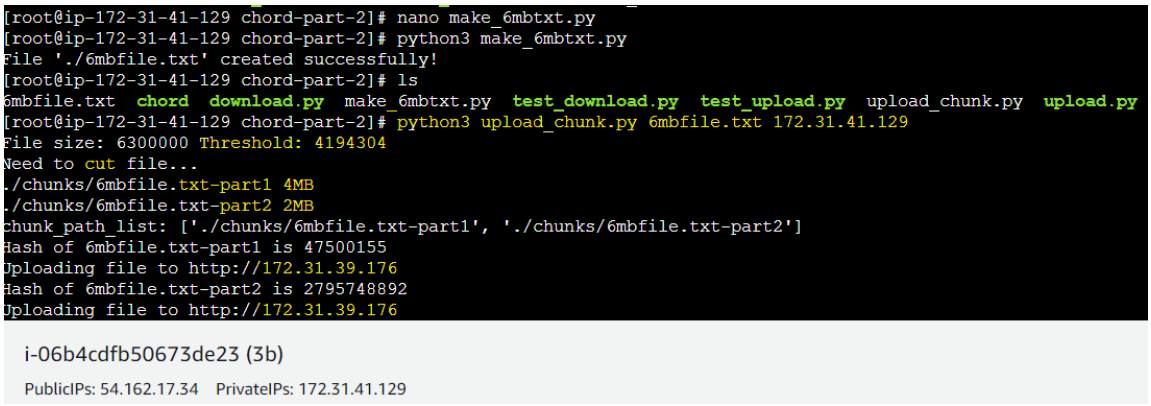
2. 於是把 replica.txt 放到 p3a 機器上，等待1分鐘後，p3b 去機器查看備份。



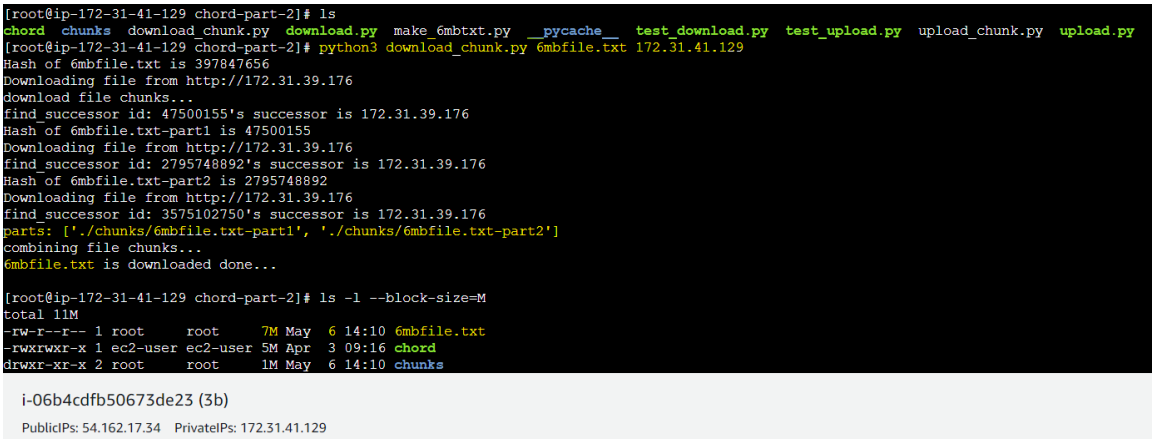

```
python3 act_findsucc.py 172.31.41.129 replica.txt #1
touch files/replica.txt #2
ls files
```

▼ File chunks demo

- 上傳下載可以在任意目錄下執行。上傳:



下載:




```
python3 make_6mbtxt.py #建立大檔案
python3 upload_chunk.py 6mbfile.txt 172.31.41.129 #上傳
```

```
ls ./files -l --block-size=M #去存放的機器檢視檔案

rm 6mbfile.txt #移除剛剛上傳的原始檔
python3 download_chunk.py 6mbfile.txt 172.31.41.129 #下載
```

打包image前需要改的檔案

- init_script.py
- 所有 py 檔
- Auto-Scale-Part2-0425