

# 29. 案例篇：Redis回應嚴重延遲，如何解決？

2020-8-13 Freddy Fan

http://192.168.0.10:10000/



# Flask

web development,  
one drop at a time



# redis

VM1: 案例应用


curl

VM2: 客户端

# 案例情況

撈個資料超級慢！！

```
FreddyFande-MacBook:~ ChienMingFan$ curl -w "%{time_connect} + %{time_starttransfer} = %{time_total}\n" -o /dev/null http://192.168.0.15:10000/get_cache
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
100 64970  100 64970    0     0  8437      0  0:00:07  0:00:07 --:--:-- 14583
0.032290 + 7.694920 = 7.700261
```



實際跑跑看吧！！

# DEMO小筆記

```
while true; do curl -s -w "%{time_connect} + %{time_starttransfer} = %{time_total}\n" -o /dev/null  
http://192.168.0.16:10000/get_cache ; done 循環跑curl
```

top 看看是否很忙

iostat -d -x 1 看看硬碟讀寫速度

pidstat -d 1 查看IO操作的程序 確認操硬碟的程式是不是 redis

strace -f -T -tt -p 1773 追蹤一下程序

lsof -p 1773 查看操作對象

docker exec -it redis redis-cli config get 'append\*' 看redis設定

strace -f -p 1773 -T -tt -e fdatsync 看看寫的情況 喔喔速度超快

strace -f -T -tt -p 1773 刷刷刷～

可以看到SADD

PID=\$(docker inspect --format {{.State.Pid}} redis-app)

nsenter --target \$PID --net -- lsof -i 看網路應用狀態

```
while true; do curl -s -w "%{time_connect} + %{time_starttransfer} = %{time_total}\n" -o /dev/null  
http://192.168.0.16:10000/get_cache_data ; done 跑不執行SADD的curl
```

# Redis小知識

## 持久化資料儲存的設定

持久化模式	说明	优点	缺点
rdb模式	定期将redis当前内存数据快照备份到硬盘	redis重启时恢复速度快	由于备份不宜频繁，会导致系统异常宕机时，redis大量数据丢失
aof模式	将redis执行的命令每隔1s批量同步到文件中	机器异常宕机时，最多丢失1s数据	由于保存的是命令，会导致保存很多冗余数据，文件过大(这时候恢复起来相对rdb会慢很多)

## AOF參數設定

`appendfsync [ no , everysec , always ]`

`no` : redis不主動同步, 系統預設30秒

`everysec` : 一秒

`always` : 每一個動作都同步

# Redis : SADD

加入元素進一集合體內

```
127.0.0.1:6379> SADD HI "111"  
(integer) 1  
127.0.0.1:6379> SADD HI "222"  
(integer) 1  
127.0.0.1:6379> SMEMBERS HI  
1) "111"  
2) "222"  
127.0.0.1:6379> 
```

## 延伸討論

- 軟體卡住會先重啟嗎？呂布治百病？
- Redis的問題比較傾向開發者問題居多，同意嗎？
- 可以把RDB的資料直接轉換成KEY-VALUE方式嗎？

把所有資料塞在同一個key內

**A String value can be at max 512 Megabytes in length.**



## 補充

redis 持久化流程：<https://www.lbbniu.com/7014.html>

兩種持久化分析：<https://kknews.cc/zh-tw/code/mrkbr8z.html>

redis一致化

: <https://ithelp.ithome.com.tw/articles/10224221>

Big O :

<https://rob.conery.io/2019/03/25/wtf-is-big-o-notation/>