Apache Tuning

小弟最近管理某單位的伺服器群，最近遇到了連線數的瓶頸，很多時候網頁的回應是慢得可以，或是直接停止回應，後來進入系統觀察時發現 CPU 和 Memory 也沒有達到巔峰，因此決定動手調整 Apache。

加入 mpm worker 模組

Apache 2.0 許多功能和改進，編譯時要開啟 worker mpm 模組，否則為了和 Apache 1.3 相容會自動選擇 prefork 模式。如下圖所示，紅色的 --with-mpm=worker 是重要的參數值。

```
steven $ ./configure --prefix=/usr/local/httpd --enable-isapi --enable-file-
cache --enable-echo --disable-charset-lite --enable-charset-lite --enable-cache
--enable-disk-cache --enable-mem-cache --enable-example --enable-case-filter --
enable-case-filter-in --enable-dumpio --enable-auth-ldap --enable-ext-filter --
enable-deflate --enable-log-forensic --enable-logio --enable-mime-magic --
enable-headers --enable-proxy --enable-proxy-connect --enable-proxy-ftp --
enable-proxy-http --enable-ssl --enable-optional-hook-export --enable-optional-
hook-import --enable-optional-fn-import --enable-http --enable-cgi --enable-cgid
--enable-speling --enable-rewrite --enable-so --with-suexec-uidmin --with-
suexec-gidmin --with-suexec-logfile --with-suexec-safepath --enable-static-
htpasswd --enable-static-htdigest --enable-static-rotatelogs --enable-static-
logresolve --with-mpm=worker
```

讓你的機器發揮到極致

在一個高等級的硬體設備下（Intel 64 位元 CPU x 2、2G RAM），以下的設定在同一時間可以處理 5000 個連線（其實還可以容許更多）。

```
root # vi /usr/local/httpd/conf/extra/httpd-mpm.conf
-----------------------------------------------------
<IfModule mpm_worker_module>
# Apache 啟動時先開啟 3 個 Threads
StartServers 3
MaxClients 5000
# 最大的連線數
ServerLimit 50
MinSpareThreads 50
MaxSpareThreads 200
ThreadLimit 200
ThreadsPerChild 100
MaxRequestsPerChild 200
</IfModule>
-----------------------------------------------------
root #
```

關於 mpm worker 的設定技巧，可以參考 Apache 手冊。

- StartServers
- MaxClients
- ServerLimit
- MinSpareThreads
- MaxSpareThreads
- ThreadLimit
- ThreadsPerChild
- MaxRequestsPerChild

減少不必的等待

當然，為了效能，我建議關閉 DNS 的查尋和提高 MaxKeepAliveRequests 的選項並且把 Timeout 值調低。

```
root # vi /usr/local/httpd/conf/extra/httpd-default.conf
-------------------------------------------------------
# 連線超過 60 秒失敗就重試
Timeout 60
# 開啟 KeepAlive
KeepAlive On
# 設定同一時間可容許的 KeppAlive 量
MaxKeepAliveRequests 5000
# KeepAlive 多久要自動 Timeout 掉
KeepAliveTimeout 3
# 關掉那費時的 DNS 查尋
HostnameLookups Off
-------------------------------------------------------
root #
```

連線設定需要考量到很多因素，在早期大部份的網路使用者頻寬都不夠大，所以一旦你設定太短的 Timeout 值就有可能會馬上讓使用者需要再次重新連線。但是現今網路使用者大部份以 ADSL 上網，所以等待時間應該不用這麼的久。

小弟所調整這些情況，其客戶端大部份為學術單位、政府單位，所以對這些大頻寬的使用者，太久的 Timeout 值有可能會變成一種不必要的資源浪費。

note:
為什麼 Timeout 值要縮短，在一個大流量的網路服務（良好的硬體設備，高頻寬的網路速度）在同一時間是需要能夠服務更多的使用者，如果 Timeout 太高，那麼在連線佔滿時下一個連線請求就要等到有連線結束掉才能夠被服務；相反的若設定太低那麼就要常常重新連線。不過重新連線一般的使用者並不會有太大的感覺（TCP/IP 的三向交握很快），所以不要讓伺服器一直在等待沒有請求的連線。
當一個連線被建立之後，使用者很有可能會在短時間之內又提出新的請求，所以為了避免再次的連線，就會在本次要求結束後，保持連線 KeepAliveTimeout 秒，若是超過了 KeepAliveTimeout 秒，其 Timeout 就會轉交給連線逾時，也就是 Timeout 秒之後，會自動斷掉連線。

讓你的 Apache 更靠近 CPU

在一台設備良好的伺服器上，上面如果只有一項服務的話，那麼就應該把大部份的資源都留給它，比方說 Web 服務。

在 Linux 上，根據啟動時加以調整 nice 值也可以讓 Apache 的處理速度快一點。

nice 的調整範圍是在 -20～+19 之間，如果不設定的設所有服務都是在 0。nice 的關念，就是它的值越低，就離 CPU 越近，那麼就跟容易取得 CPU 的資源，相反的在正數越高（+1～+19）越不容易得到回應。

啟動 apache 時就把 nice 值調到 -10

```
root # nice -10 /usr/local/httpd/bin/apachectl start &
```

如果系統已經在啟動的話，可以使用 renice 來重新設定其 nice 值。找出 apache 的 PID，並重新設定

```
root # netstat -ntulp | grep httpd
tcp 0 0 0.0.0.0:80 0.0.0.0:* LISTEN 19249/httpd
root # renice -10 19249
```

當你調整完畢之後，可以使用 top 指令查看 NI 欄位的變化。

取消不必要的服務

把不要必要的服務全部都關掉以便適出更多的可用資源讓 Apache 使用，當系統安裝時，大部份的系統建置者通常會把所有的套件都裝上去，以免未來要使用的功能不齊全，當然這是我們無法阻止的，但是我們卻可以控制系統的服務。

如果你使用 Redhat/Fedora 系統 Linux OS 的話，可以使用 chkconfig 來看看系統啟動時到底有那些服務被開啟。

```
root $ chkconfig --list
~ 以上略 ~
rawdevices 0:off 1:off 2:off 3:off 4:on 5:on 6:off
acpid 0:off 1:off 2:off 3:on 4:on 5:on 6:off
ipchains 0:off 1:off 2:on 3:off 4:on 5:on 6:off
iptables 0:off 1:off 2:on 3:off 4:on 5:on 6:off
crond 0:off 1:off 2:on 3:on 4:on 5:on 6:off
anacron 0:off 1:off 2:on 3:off 4:off 5:off 6:off
lpd 0:off 1:off 2:on 3:off 4:on 5:off 6:off
xfs 0:off 1:off 2:on 3:off 4:on 5:on 6:off
ntpd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
portmap 0:off 1:off 2:off 3:off 4:on 5:on 6:off
xinetd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
autofs 0:off 1:off 2:off 3:off 4:on 5:on 6:off
nfs 0:off 1:off 2:off 3:off 4:off 5:off 6:off
nfslock 0:off 1:off 2:off 3:off 4:on 5:off 6:off
identd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
~ 以下略 ~
root #
```

使用 chkconfig 可以看到各服務在系統那個 system level 時會被關閉或開啟。如果你的機器只有單純的服務網頁服務，那麼就不應該要有 NFS、telnet、ldap、samba、lpd、cups、mysql 或會佔用大量記憶體及 CPU 資源的程式，尤其是像資料庫服務。

以下示範如何把 MySQL 在開機時不啟動。

```
root # chkconfig mysqld off
```

如果 MySQL 正在執行，要把它關閉的話。

```
root # service mysqld stop
或
root # /etc/init.d/mysqld stop
```

重新編譯 Apache!

如果你對於系統廠商所附的套件不滿意，或是想要轉到新版本的 Apache 時，那麼就要編譯一下 Apache，在編譯時可能需要注意一些事項：

- 不要把所有的模組都編上去
- 確認你的 GCC 版本

不要把所有的模組都編上去

如果一開始就決定要使用 JSP 開發程式，那麼就不需要把 PHP 模組都套進來；若不需要使用到 LDAP 認證，那也不需要把 LDAP 的模組也一起編進來，總之，在編譯你的 Apache 之前，那麼就應該要先確認環境狀況而不是一頭腦的全都加進去。

確認你的 GCC 版本

說真的，小弟對於程式並不是很熟，所以無法明確的指出 GCC 版本對於編譯出來的程式有什麼樣的結果。不過可以證實的是，有些新版本的 GCC 在編譯時，可以讓程式更小執行更快。

調校時應該注意的事項

當你開始動手調整之前，應該要考慮主機的能力，比方說在記憶體只有 256 MB 的主機要去服務上千個連線這似乎有點勉強，你應該要注意所設定的數值不會超過主機所能負荷的能力，比方說吃掉 90% 的記憶體，那麼到時候可能會得到反效果。

在轉換 Apache 版本之前，最好是可以在一台測伺主機上編譯一次，若有問題時可以有時間解決，否則一旦在正式上線的主機上做這種實驗又沒有備份的情況下，可能會有無法預期的結果出現。

正式在執行你的新 Apache 之後，你應該要特別注意系統的負荷、主機資源的使用、效能是否如預期。如果結果不是在預期之內，應該要試者再做更細微的調整，請仔細閱讀 Apache 網站上的手冊，有時候一些小小的調整可以達到不錯的效果 :)。

參考資料

這篇文章是小弟在工作時實戰的心得，因此會有很多疏忽的地方，如果有錯也請大家不吝指教。

以下為一些參考資料：

- Apache 錦囊妙技（O'Reilly, ISBN: 986-7794-31-1）
- Apache MPM worker
- Apache 2.0 性能優化（簡體文）

For more articles, please visit http://www.l-penguin.idv.tw/~steven/

作者：廖子儀 (Tzu-Yi Liao)
Certified：LPIC Level I、LPIC Level II、RHCE
E-mail：steven@ms.ntcb.edu.tw
Web site：Steven's Linux Note (http://www.l-penguin.idv.tw/~steven/)