

# 01 如何學習 Linux 效能最佳化？

Rick Hwang  
2020/04/23

# 用詞對照 (英文:簡中 <-> 繁中)

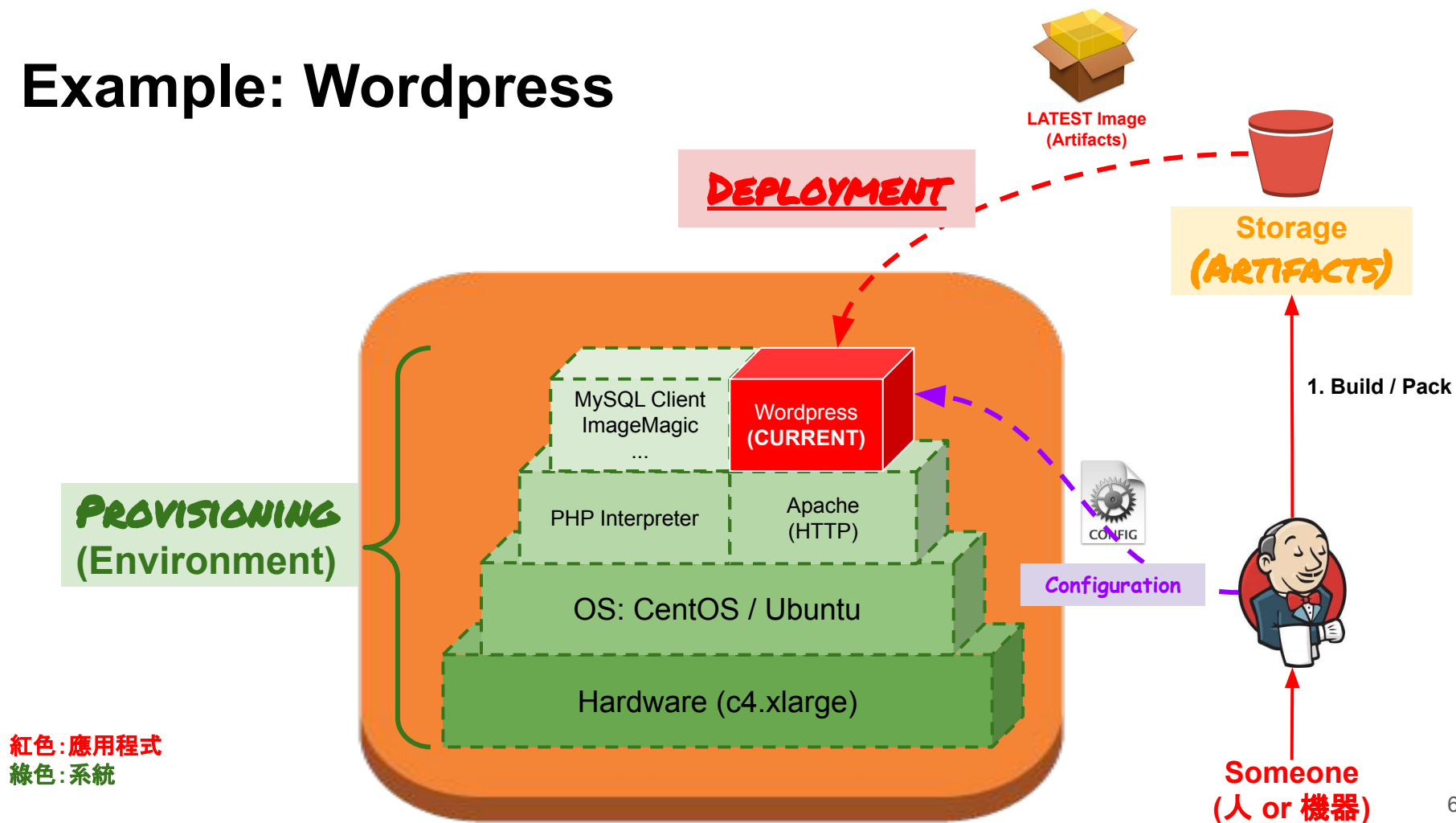
- Optimization: 優化 ← → 最佳化 <https://lds.guru/g7c4vg>
  - 改善 (Improvement)
- Performance: 性能 ← → 效能
- Components: 組件 ← → 元件、模組 (Module)
- Application: 應用程序 ← → 應用程式, 簡稱 AP
- Programming Language: 編程語言 ← → 程式語言
- Latency: 延時 ← → 延遲

你不需要瞭解**每個元件**的所有**實現細節**  
只要能理解它們**最基本的工作原理**和**協作方式**

你不需要瞭解**每個人的所有心裡的想法**  
只要能理解他們**最基本的工作原理和協作方式**

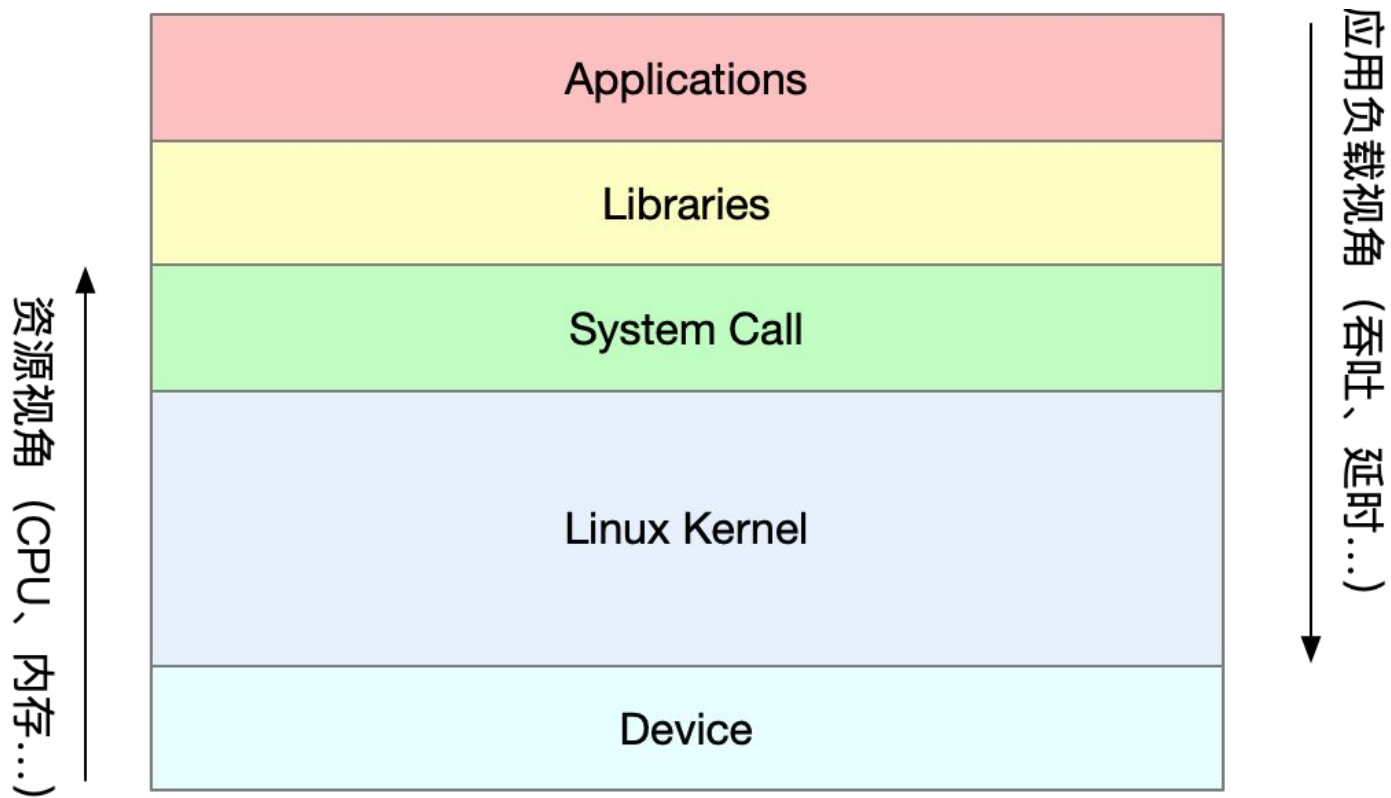
只要你理解了**應用程式**和**系統**的少數幾個基本原理  
再進行大量的**實戰練習**，建立起整體效能的全局觀

# Example: Wordpress



# 效能指標是什麼？

- 高並發 (High Concurrency) 、響應快 (Quick Response)
- 吞吐 (Throughput) 和 延遲時 (Latency)
- 從**應用負載**的視角：直接影響了產品終端的用戶體驗 (End User)
- **系統資源**的視角：資源使用率、飽和度






# 效能問題的本質

1. 系統資源已經達到瓶頸
2. 請求的處理卻還不夠快
3. 無法支撐更多的請求



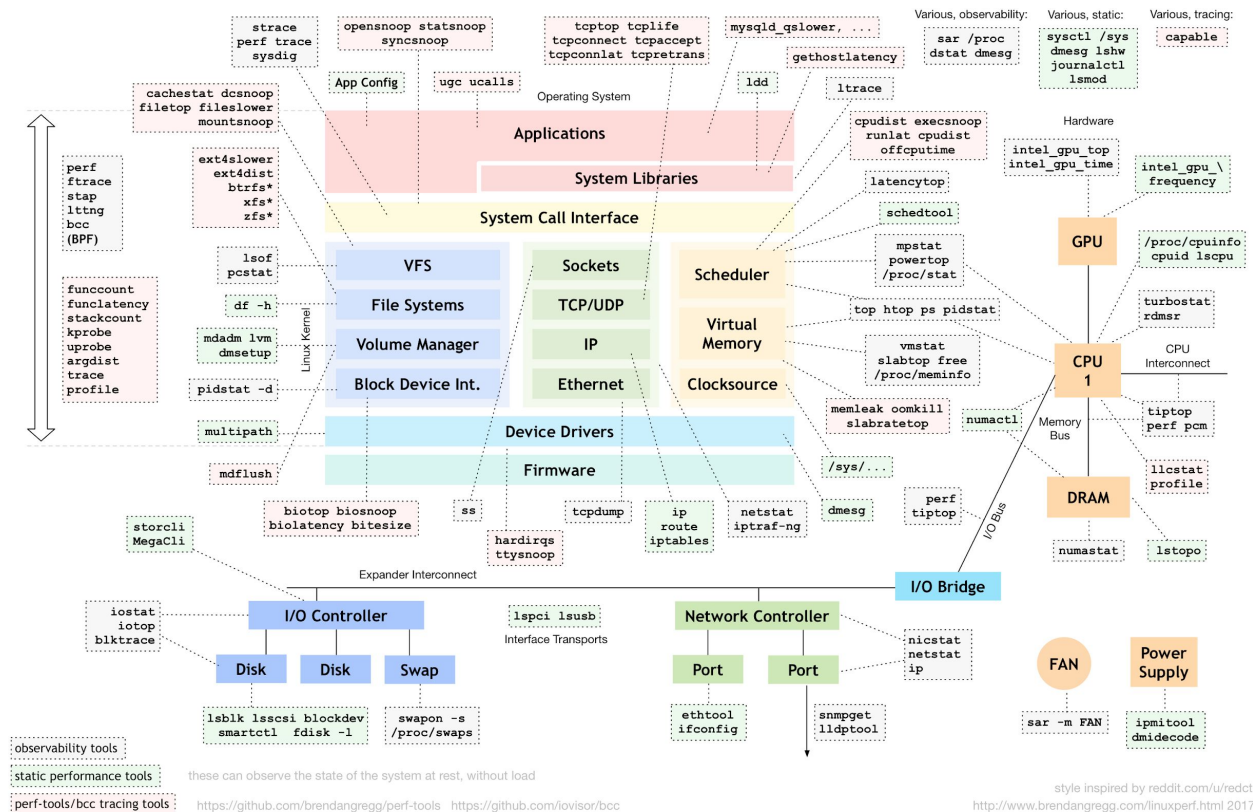
錢能解決的問題？

# 效能分析的步驟

1. 選擇**指標**, 評估**應用程式**和**系統效能** → SLI (Service Level Indicator)
  2. 為**應用程式**和**系統**設置效能目標 → SLO (Service Level Objective)
  3. 進行效能基準測試 → **Benchmark、Capacity**
    - a. 如何量測系統的容量？(壓測)
  4. 效能分析定位瓶頸 → Bottlebeck
    - a. 系統發生異常時, 第一時間如何快速止血？
  5. 改善 (優化) 系統和應用程式 → 重構
  6. 效能監控和告警
- 

# 學這個專欄需要什麼基礎

1. 瞭解 Linux 常用命令的使用方法
2. 知道怎麼安裝和管理軟件包
3. 知道怎麼通過程式編程語言開發應用程序等



## BPF Performance Tools: Linux System and Application Observability

# 怎麼學更高效？

技巧一：雖然系統的原理很重要，但在剛開始一定不要試圖抓住所有的實現細節。

技巧二：邊學邊實踐，通過大量的案例演習掌握 Linux 性能的分析 and 優化。

技巧三：勤思考，多反思，善總結，多問為什麼。

# 案例分享

- Wordpress 網站
  - ELB、EC2 (AutoScaling)、RDS
  - 沒有 Cache (Redis)
- 現象：
  - ALB Latency 很高
  - RDS connection 爆量
  - 減少機器後，無效
  - 新增機器也無效



Rick Hwang 大概是這樣：

## Rick 的分析思路

1. 看到全貌
2. 順著路 (Request) 找到斷點、熱點
3. 了解每個點的特性 (平常要做的功課)

系統發生異常時, 第一時間如何快速止血?

1. 把每個 service 的 downstream / upstream or dependency (外部依賴) 搞清楚
2. 確立每個來回的通訊協議 / 模式: http / tcp / grpc / .... sync or async
3. 確立這些都有蒐集系統的 metrics (cpu/mem/disk/tcp/io ..), log 吐去哪 ...
4. 把這條路徑上的瓶頸找出來. 對口的水管可能大小不一, 例如 t2.nano 對 c5.large 的不對等的水管 ...
5. AP 本身的特性: CPU-Bound, Mem Bound, IO Bound ...

通常到 3 能確認, 問題就七七八八了, 很常是 AP 本身沒寫好、沒處理好 Exception、資源吃太多炸鍋。但很多時候, 是連 1) 有哪些都搞不清楚, 或者不知道要找誰。

然後可以把 1~5 都搞清楚了, 再來討論是否有 #效能議題 ... 像是加 Cache 之類的緩衝 ...

通常 1) 搞清楚後, 會發現整個架構很詭異, 但是每天卻要跟他共生共存 XDD

以架構的演進原則來看, 勢必要適度的收斂, 最好是把網狀結構, 往星狀結構走, 否則整個維運會很難搞的。

好吧, 我去年有講啦, 在 P66 ... 連 1) 都搞不清楚, 本身就是災難。根本不用談什麼架構 ...

<https://rickhw.github.io/...../DevOpsDaysTaipei2018...../>

讚 · 回覆 · 2週 · 已編輯



# 問題與討論

- 單一台機器的效能與多機器(服務)效能的分析, 有什麼共通性?



而言永遠記得它是香蕉油，因為每次我喝過摻有此種香味的東西後，我都會再記得一次。這是經過無數次重覆，從儲存式的記憶到反射式的記憶。同理，你一定也忘了那些大學聯考曾考過的命題，原因是你根本從未再覆習這些東西了。

因此「練習」這件事，對腦部的運動而言，根本是在做「重覆」的記憶工作罷了，有沒有一句背好的 II - V - I 的句子，是在你不由自主的彈奏下，自然的「蹦」出來？真正的 Real Time Playing 所用的部份，大多是反射式記憶，而非昨天你才學會的 Robben Ford 的句子。

記憶與理解

反覆100次數以上

>>>>>>

不斷地練習

反射與直覺

左腦

右腦

不要陷入練習上的迷思。搞不清楚現在練習的部分是什麼以及現在練習到「什麼程度」。「左腦嗎」？還是已經「右腦」了！需知，由左腦到右腦，是沒有捷徑的，有差別的是反覆的次數（它反映你的天賦）。