105-1 CSIE System Programming

Report for HW4

資工二 B03203004 潘廣霖

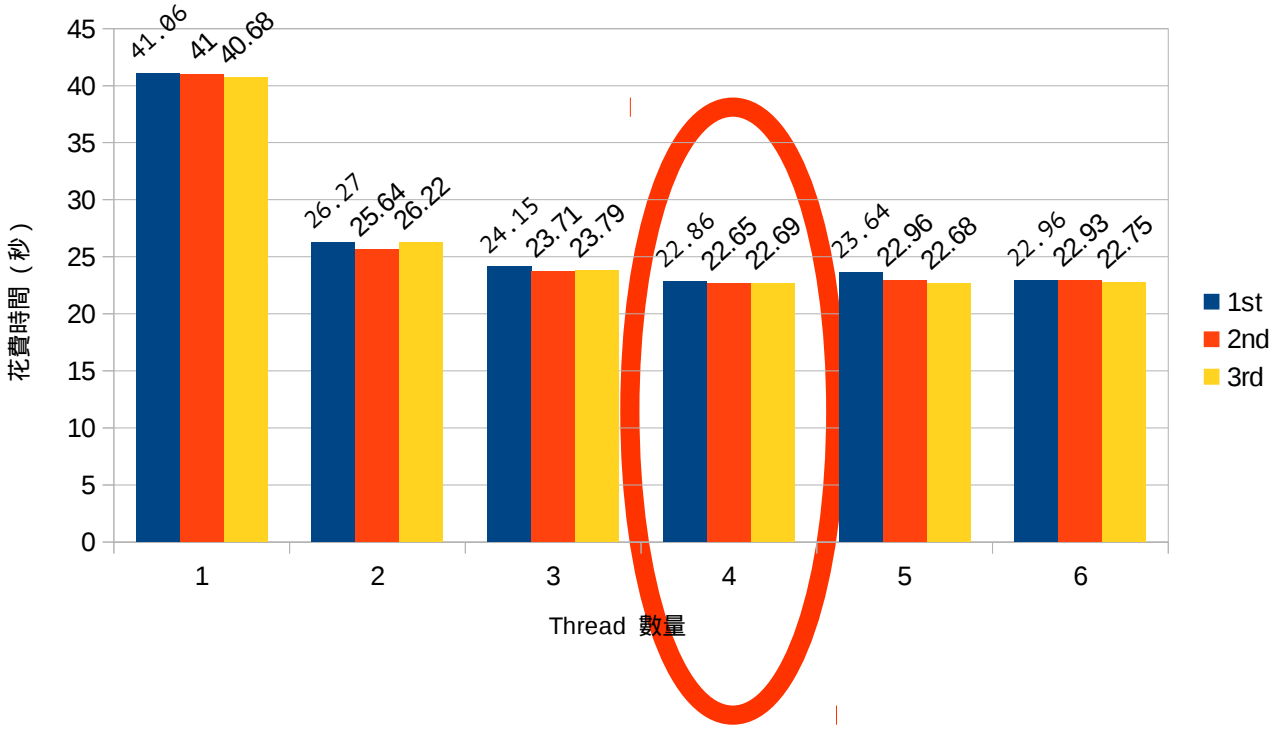
2017-01-17

1. thread 開在哪裡？如何分工？

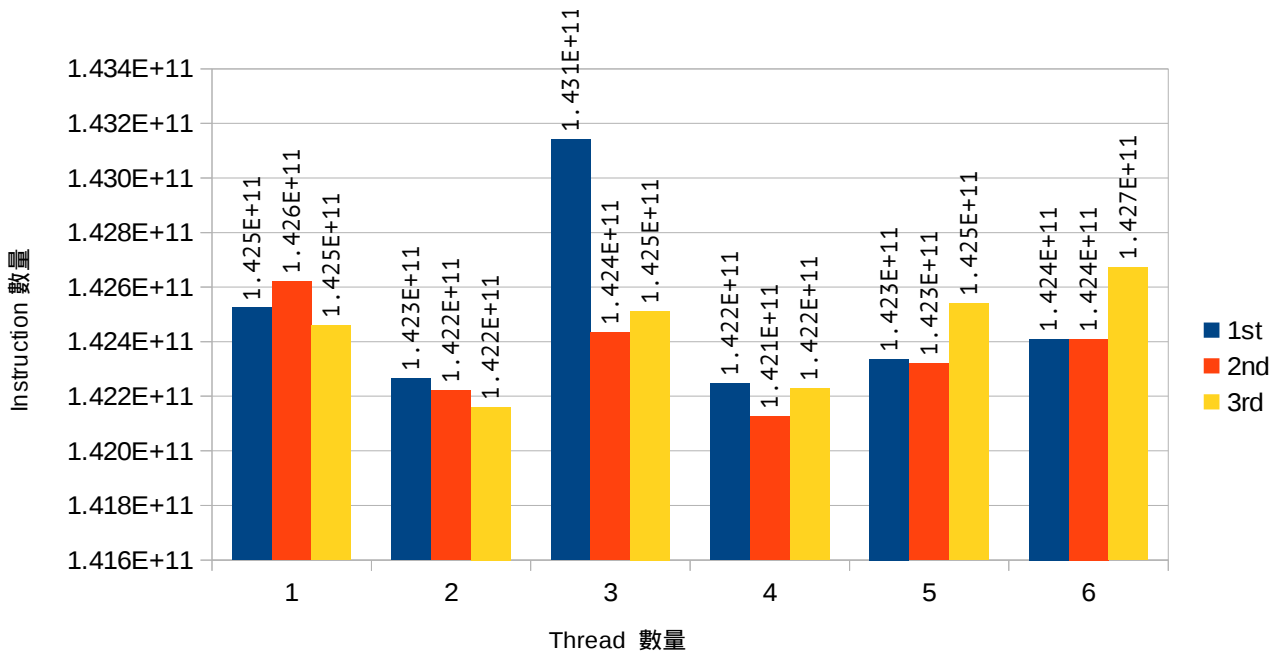
開在 train 的階段，讓不同的 threads 分別平均 train 一定數量的樹，並將這些樹的根節點填回陣列裡，形成森林。用 mutex 做樹節點的記憶體管理。

2. thread 數量與時間的比較。

測試環境為原生 Ubuntu 16.10 64bits，CPU 為 Intel i5 2.50GHz x 4. L1/L2: 128KiB/512 KiB
本圖計算 train 500 顆樹並進行測試的執行時間，每個項目進行 3 次測試。執行最快的 thread 數量與 CPU 核心數相同，而且隨著 thread 數量增加，邊際效益遞減。預期繼續增加數量應該可以獲得更好的效能，但是反而收效甚微，甚至時間還增加了，推測是程式存取大量資料造成的 overhead 還有樹節點大量 interleave 造成 cache miss 太嚴重。

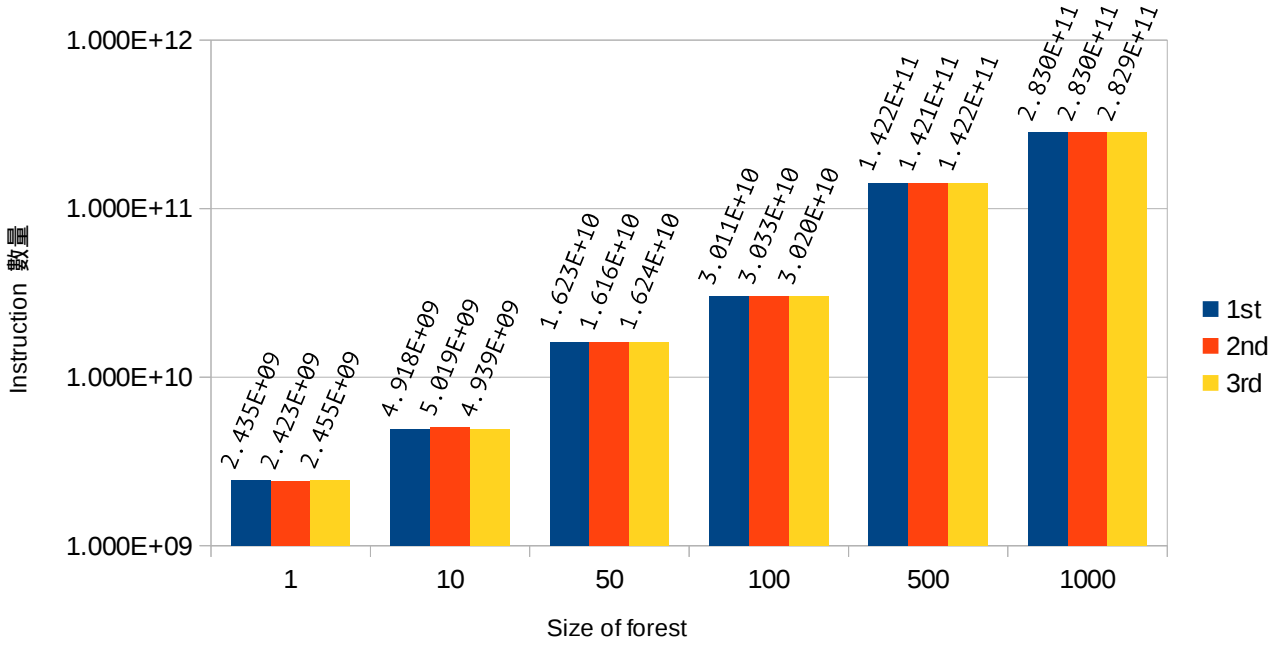


3. thread 數量與 instructions 數量的比較。



本圖測試條件及使用的指令 (perf) 與上圖完全相同。數值分佈範圍很小，每個 thread 看不出明顯規律，只能勉強說 1 和 3-threads 的 instruction 數略大於 2 和 4-threads 時。

4. 樹的數量與 instructions 數量的比較。



此圖已取對數座標，以 4-threads 進行測試。可以約略看出 instruction 的數量約略和樹的數量成正比。這是因為扣掉讀入測資、取用記憶體、初始化必要的資料結構等等常數的 instruction 後(這個值大略和 1 棵樹的情況接近) 所花費時間漸進於常數倍的樹的數量。(若把排序當作整個演算法的瓶頸的話，這個漸進複雜度可能是 O(n log n)，但實際要考慮的因素很多，而且規模還不夠大，很難單純藉由分析整體時間來得知瓶頸，需要更細緻的效能分析。)

5. 其他。

關於正確率：一開始發現測資全部填零就上 80% 了，來說說在 tune 參數之餘的發現。因為樹如果太深程式會跑不完，最簡單的方式是剪枝 (prune) 然後在結尾判剩下群的主要 label 來概括這組資料。但如果在還有幾千筆資料下貿然剪枝，效果太粗糙等於沒做，造成每次的結果會在 75~85%上下波動，就算森林很大 (約 1000 棵樹)，準確率也上不去。我把每次切割的 impurity 印出來，發現若選到的測資很雜亂其實不管選哪個切 impurity 都很高 (0.151~0.156 左右)，因為公式特性的關係，只挑最低 impurity 的切點，往往每次只會在兩端切一兩筆資料下來，嚴重造成分群效果不彰，於是最顯著改進的方式是每次切割的時候捨棄切下兩群數量相差太多倍數的切法(假設分成 X:Y 的兩群，要求 $p=\frac{X}{X+Y}\times\frac{Y}{X+Y}$ 不能太大，比如我想限制最差情況下樹的層數在 50 左右，取 $\log_p n = 1/50$，在 $n = 10{,}000$ 時 p 約 0.1)。這方法的缺點是會在某些極端情況下明明可以切節點下來卻放棄不切，所以是個 trade-off。一個可能的改進是可以紀錄下「次差」的切法，在沒辦法切的情況下放鬆條件。