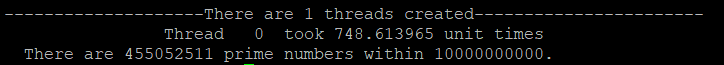
**B0521229 林威廷**

**平行程式設計-Training5**

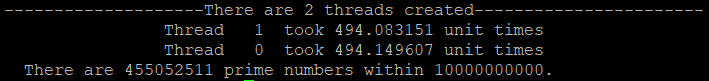
**Results of Implementation**

1 Thread (Base)



Time:748.6139 (Base)

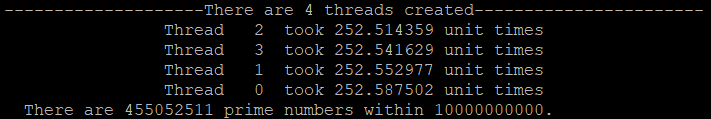
2 Threads:



Time:494.1496

Speedup factor=748.6139/494.1496=1.515

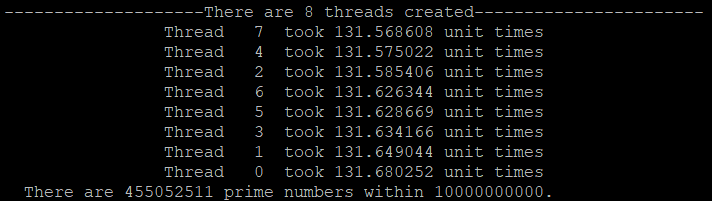
4 Threads:



Time:222.5875

Speedup factor=748.6139/222.5875=3.363

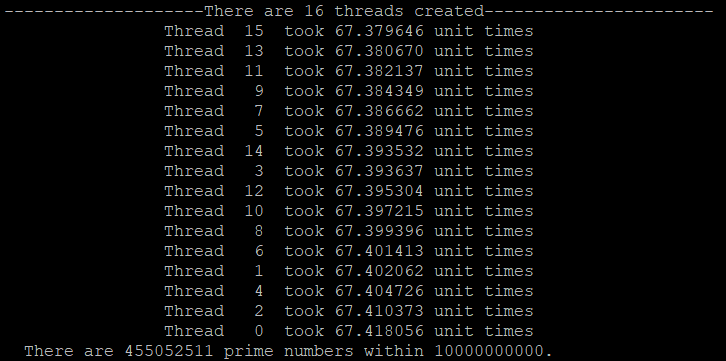
8 Threads:



Time:131.6802

Speedup factor=748.6139/131.6802=5.685

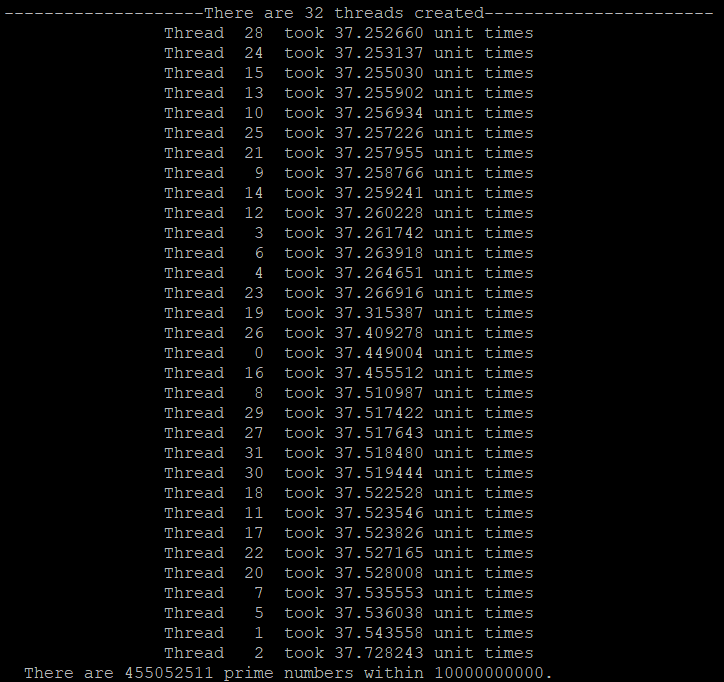
16 Threads:



Time:67.4180

Speedup factor=748.6139/67.4180=11.104

32 Threads:



Time:37.7282

Speedup factor=748.6139/37.7282=19.842

**心得(不與MPI比較原因)**

# 本次作業的實作，與Training2 非常相像，當初Training2因為要練習使用MPI去完成找質數，無法充分利用Sieve of Eratosthenes演算法(該演算法在Training2只用了root(10000000000)個數字且sequential找) ，而本次因為使用OpenMP可以共享記憶體，如我在做完Training2所預期，可充分使用Sieve of Eratosthenes演算法，加速了非常多，但因共享變數問題要處理，需加入private保護變數，且因為負載不平衡(前面質數多)，我使用schedule的runtime type去最佳化負載平衡(因為我沒加nowait所以自動有barrier會等待找完所有質數，以至於每個threads跑出來的時間差距甚小，負載很平均)，跑出來的結果很滿意，另外，此次OpenMP與上次的MPI使用環境不同(本次有多達48顆CPU而上次只有32顆CPU且這次電腦負擔比較小)、演算法稍有差異等問題，所以我不另行比較OpenMP與MPI的結果。