資料工程

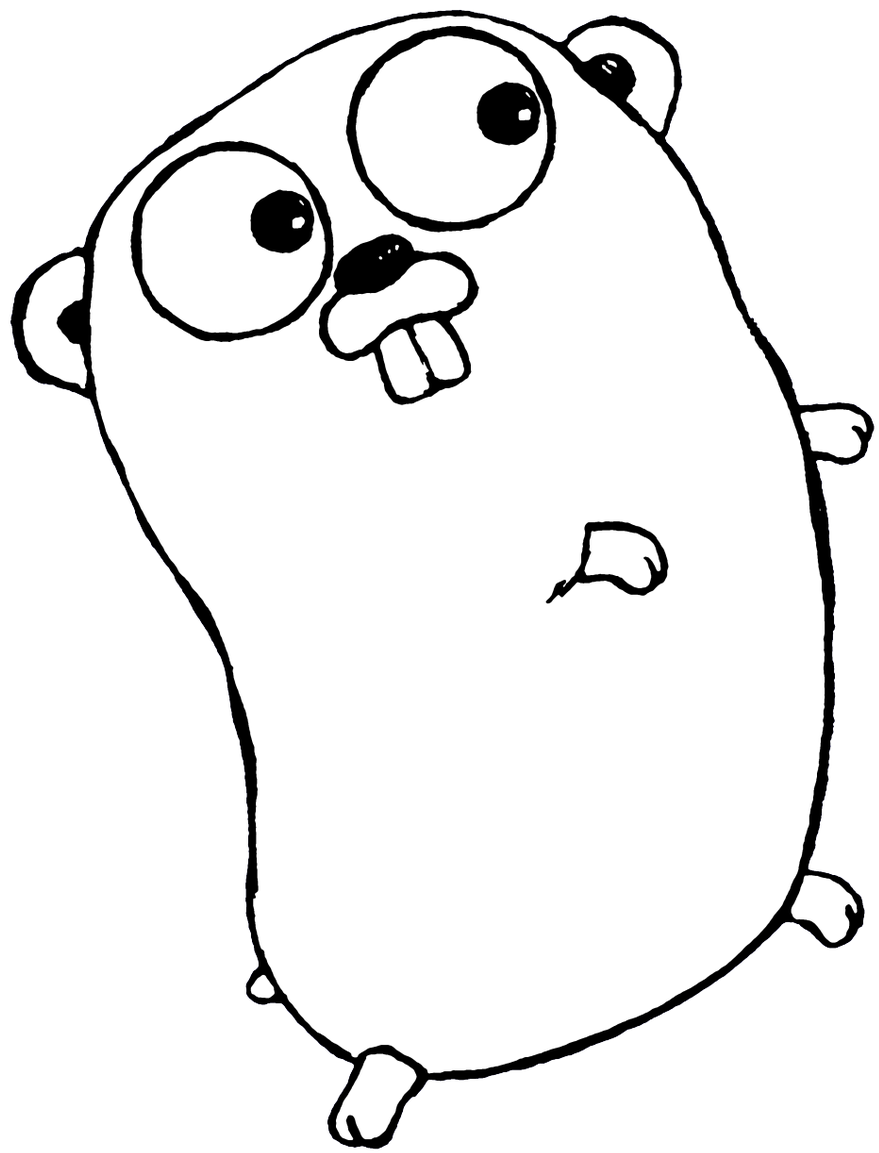
External sort

403410033
資工四
曾俊宏

實作

- External merge sort
  - 先將資料分成指定的run個數
    - 對於每個run先做sorting
      - 平行化
    - 最後利用winner tree進行合併

- 使用 golang — 真的太方便了
  - goroutine, Channel!
  - Garbage collection
  - 大推 CPU 和 memory profiler
  - 讓你更敢使用指標
  - 快速compile
  - Error message 簡單易懂



File: my_sort
Type: cpu
Time: Oct 26, 2017 at 2:05pm (CST)
Duration: 10.98mins, Total samples = 608.52s (92.35%)
Showing nodes accounting for 575.17s, 94.52% of 608.52s total
Dropped 253 nodes (cum <= 3.04s)

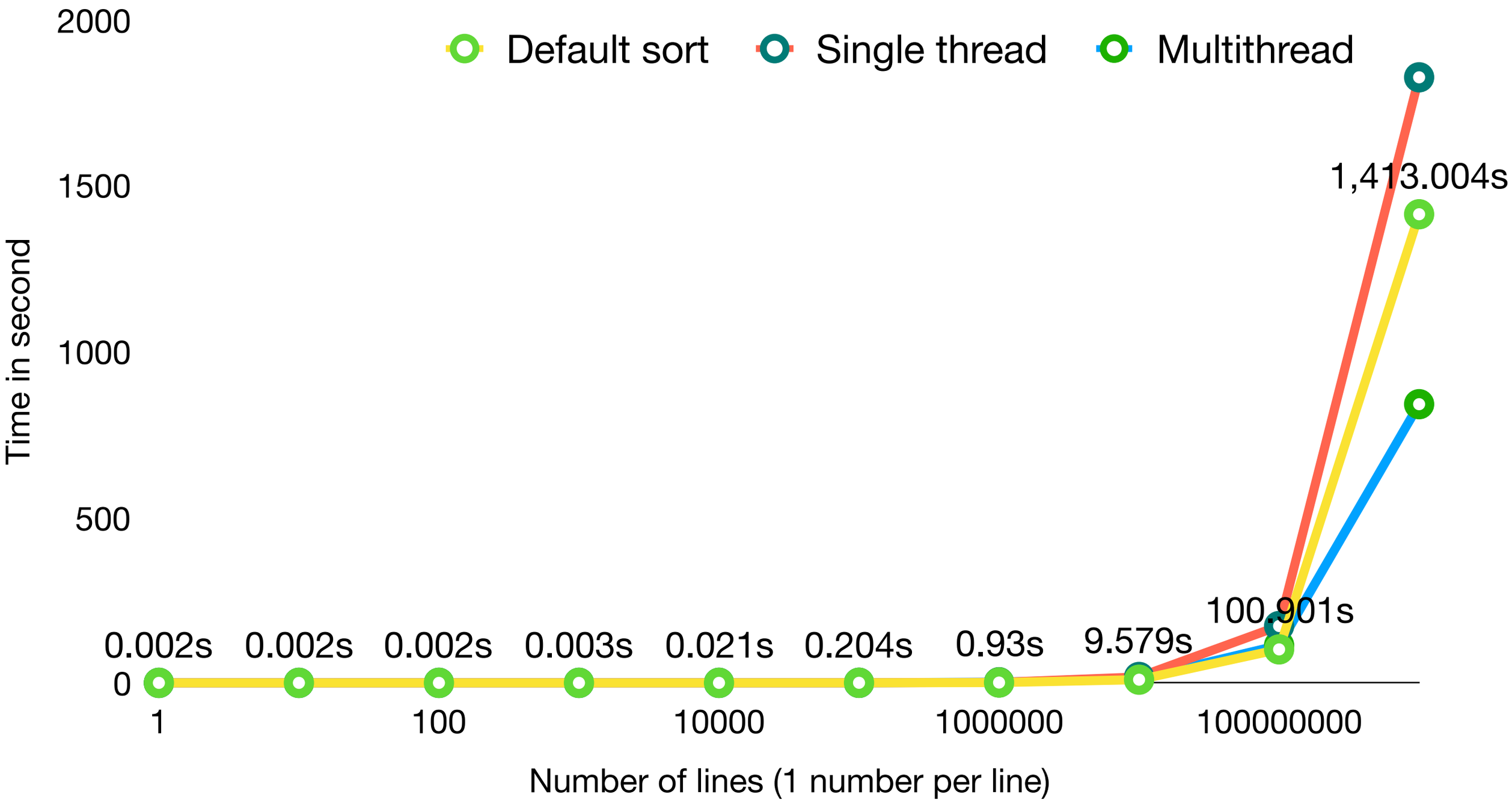
runtime main proc.go 0 of 456.21s (74.97%)
456.21s
main main external_merge_sort.go 0 of 456.21s (74.97%)
168.59s
287.62s
main mergeChunks external_merge_sort.go 1.10s (0.18%) of 168.59s (27.70%)
main splitDataIntoChunks external_merge_sort.go 2.67s (0.44%) of 287.62s (47.27%)
144.98s
10.41s
183.83s
26.65s
main (*WinnerTreeData) winnerTreePop winner_tree.go 0.93s (0.15%) of 144.98s (23.83%)
main writeTempFile external_merge_sort.go 0.68s (0.11%) of 183.83s (30.21%)
runtime concatstring2 string.go 1.68s (0.28%) of 37.06s (6.09%)
10.22s
43.13s
14.62s
12.77s
143.84s
170.31s
15.76s (inline)
35.38s
bufio (*Writer) WriteString bufio.go 4.27s (0.7%) of 22.99s (3.78%)
main (*WinnerTreeData) winnerTreeUpdate winner_tree.go 12.83s (2.11%) of 143.84s (23.64%)
sort Strings sort.go 0 of 170.31s (27.99%)
runtime concatstrings string.go 12.49s (2.05%) of 35.38s (5.81%)
runtime growslice slice.go 0.95s (0.16%) of 43.13s (7.09%)
0.96s
sort.Strings /cshome/under/u103/tch103u/go/src/sort/sort.go (170.31s)
8.55s
24.22s
93.76s
12.48s (inline)
bufio (*Writer) Flush bufio.go 0.09s (0.015%) of 8.55s (1.41%)
bufio (*Scanner) Scan scan.go 11.26s (1.85%) of 38.89s (6.39%)
main (*WinnerTreeData) winnerTreeInternalNodeUpdate winner_tree.go 54.21s (8.91%) of 93.76s (15.41%)
sort Sort sort.go 0 of 170.31s (27.99%)
bufio (*Scanner) Text scan.go 1.11s (0.18%) of 28.24s (4.64%)
runtime rawstringtmp string.go 2.50s (0.41%) of 17.14s (2.82%)
runtime typedmemmove mbarrier.go 2.24s (0.37%) of 32.50s (5.34%)
8.46s
7.47s
17.11s
170.31s
9.68s
27.13s
5.75s
1.97s
14.64s
1.78s
7.61s
1.30s
28.96s
os (*File) Write file.go 0.14s (0.023%) of 8.46s (1.39%)
os (*File) Read file.go 0.03s (0.0049%) of 7.47s (1.23%)
bufio ScanLines scan.go 3.97s (0.65%) of 17.11s (2.81%)
sort quickSort sort.go 1.92s (0.32%) of 170.31s (27.99%)
runtime slicebytetostring string.go 2.80s (0.46%) of 27.13s (4.46%)
runtime rawstring string.go 1.78s (0.29%) of 14.64s (2.41%)
runtime writebarrierptr mbarrier.go 0.14s (0.023%) of 3.46s (0.57%)
runtime bulkBarrierPreWrite mbitmap.go 2.65s (0.44%) of 28.99s (4.76%)
8.30s
7.43s
12.03s
16.59s
150.30s
2.16s
22.17s
12.86s
3.32s
23.05s
os (*File) write file_unix.go 0.08s (0.013%) of 8.30s (1.36%)
os (*File) read file_unix.go 0.14s (0.023%) of 7.43s (1.22%)
runtime indexbytebody asm_amd64.s 12.03s (1.98%)
sort insertionSort sort.go 5.50s (0.9%) of 16.59s (2.73%)
sort doPivot sort.go 39.29s (6.46%) of 150.30s (24.70%)
runtime memmove memmove_amd64.s 20.87s (3.43%)
runtime mallocgc malloc.go 18.45s (3.03%) of 42.64s (7.01%)
runtime writebarrierptr_prewrite1 mbarrier.go 1.63s (0.27%) of 26.37s (4.33%)
runtime gcBgMarkWorker mgc.go 0 of 116.91s (19.21%)
runtime _System proc.go 0 of 31.15s (5.12%)
8.22s
7.29s
1.25s
5.92s
7.32s
3.37s
1.35s (inline)
3.13s (inline)
7.20s
20.22s
116.87s
7.60s
23.51s
internal/poll (*FD) Write fd_unix.go 0.06s (0.0099%) of 8.22s (1.35%)
internal/poll (*FD) Read fd_unix.go 0.01s (0.0016%) of 7.29s (1.20%)
31.71s
5.21s
9.24s
97.36s
1.85s
sort medianOfThree sort.go 1.68s (0.28%) of 5.92s (0.97%)
7.73s
runtime nextFreeFast malloc.go 7.32s (1.20%)
runtime heapBitsSetType mbitmap.go 2.99s (0.49%) of 3.37s (0.55%)
runtime acquirem runtime1.go 4.48s (0.74%)
runtime systemstack asm_amd64.s 7.60s (1.25%) of 154.07s (25.32%)
runtime _ExternalCode proc.go 23.51s (3.86%)
1.77s (inline)
7.73s
6.84s
3.55s
0.69s
82.75s
3.69s
20.23s
33.82s
syscall Write syscall_unix.go 0.03s (0.0049%) of 7.73s (1.27%)
syscall Read syscall_unix.go 0.06s (0.0099%) of 6.84s (1.12%)
sort (*StringSlice) Less <autogenerated> 15.24s (2.50%) of 111.40s (18.31%)
sort (*StringSlice) Swap <autogenerated> 4.62s (0.76%) of 10.52s (1.73%)
runtime markroot func1 mgcmark.go 0 of 82.75s (13.60%)
runtime mallocgc func1 malloc.go 0 of 3.69s (0.61%)
runtime writebarrierptr_prewrite1 func1 mbarrier.go 1.16s (0.19%) of 20.23s (3.32%)
runtime gcBgMarkWorker func2 mgc.go 0 of 33.82s (5.56%)
7.70s
6.78s
96.16s (inline)
5.90s (inline)
82.75s
3.69s
19.07s
33.79s
sort StringSlice
runtime

測試

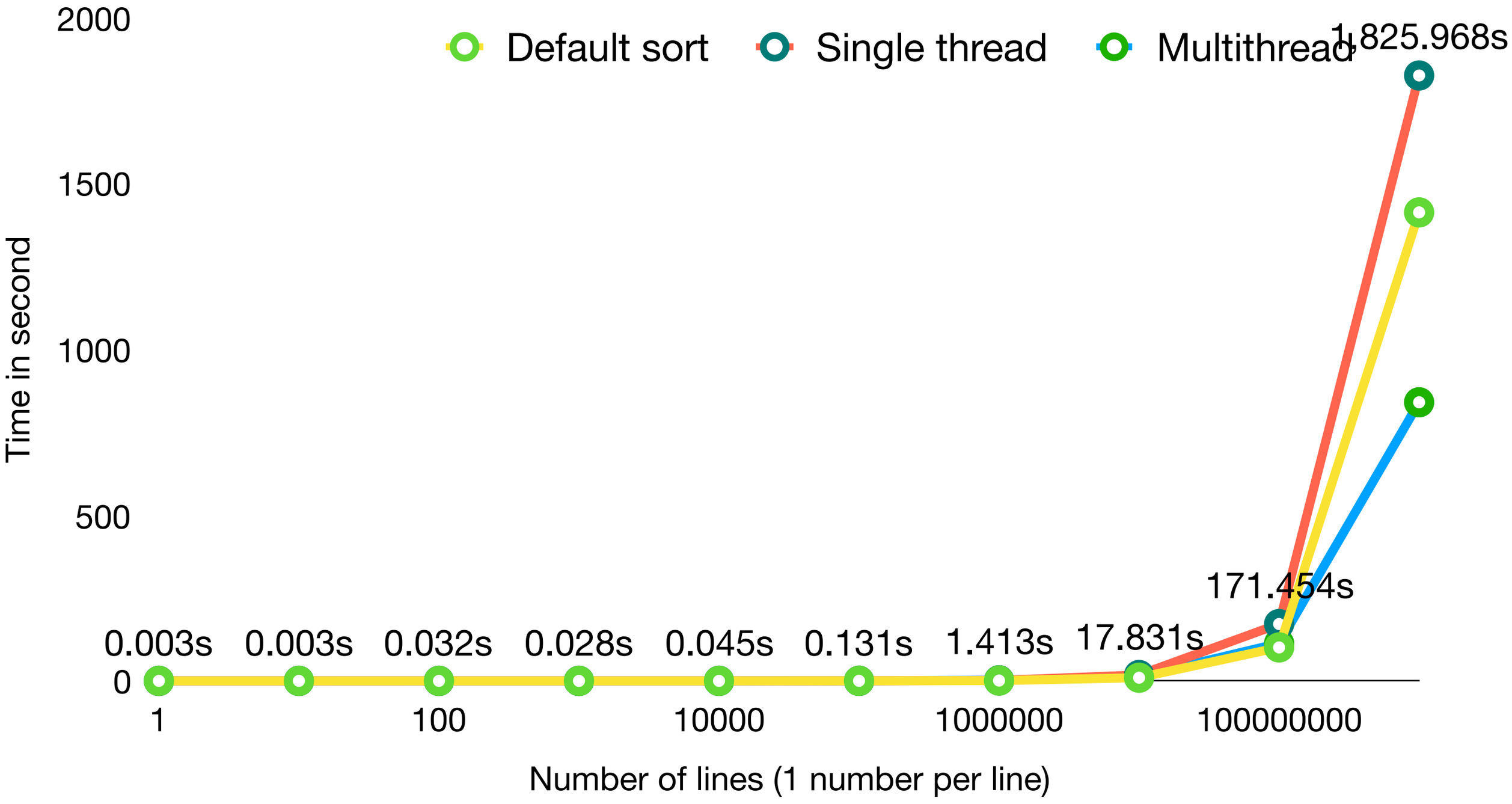
- 使用1到1000000000行數字的檔案
  - for(int ii = 1; ii <= 1000000000; ii *= 10)
- 檔案大小最大約為10GB，次大約為1GB
- 利用 sort -c 進行檢查

```
➜ 2 git:(master) ./my_sort -h
Parse command line argument
Usage of ./my_sort:
  -chunks int
        Minimal chunks to be created (default 1024)
  -cpuprofile string
        write cpu profile to file
  -d    Set true for debug mode
  -depth int
        Depth (default 4)
  -freq int
        frequency for printing debug messages (default 100)
  -i string
        The file to be sorted (default "in")
  -memprofile file
        write memory profile to file
  -o string
        The file to store sorted result (default "out")
  -p int
        Default to parallel mode (default 1)
  -pi
        Set to true to preserve the input file (default true)
  -pt
        Set to true to preserve the temporary file
  -tmp string
        The path for generated temporary files to be stored (default "/tmp")
```

效能
工學院電腦1
內建的 sorting



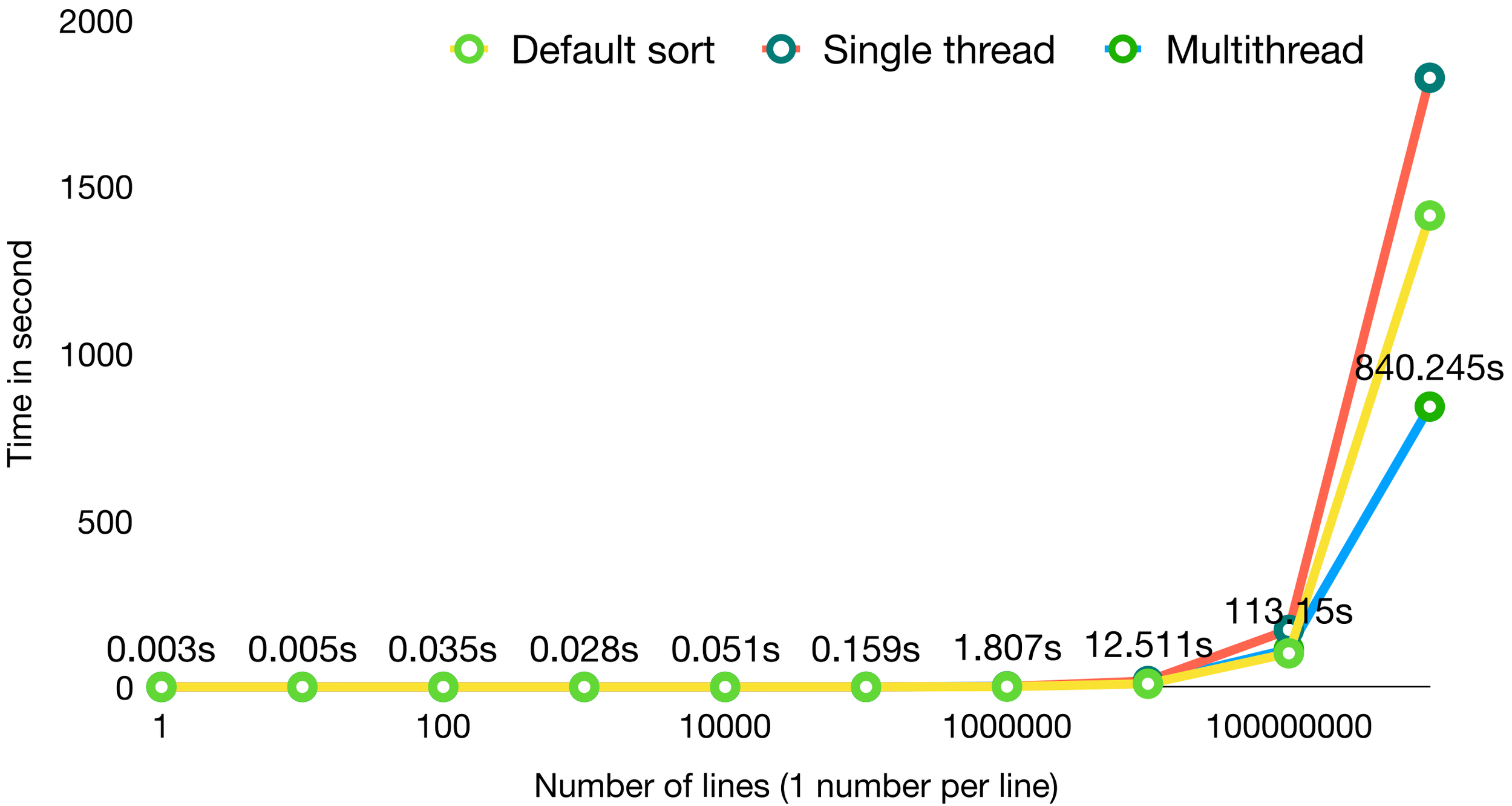
效能
工學院電腦1
單線程sorting



效能

工學院電腦1

多線程, 100 runs, depth = 6



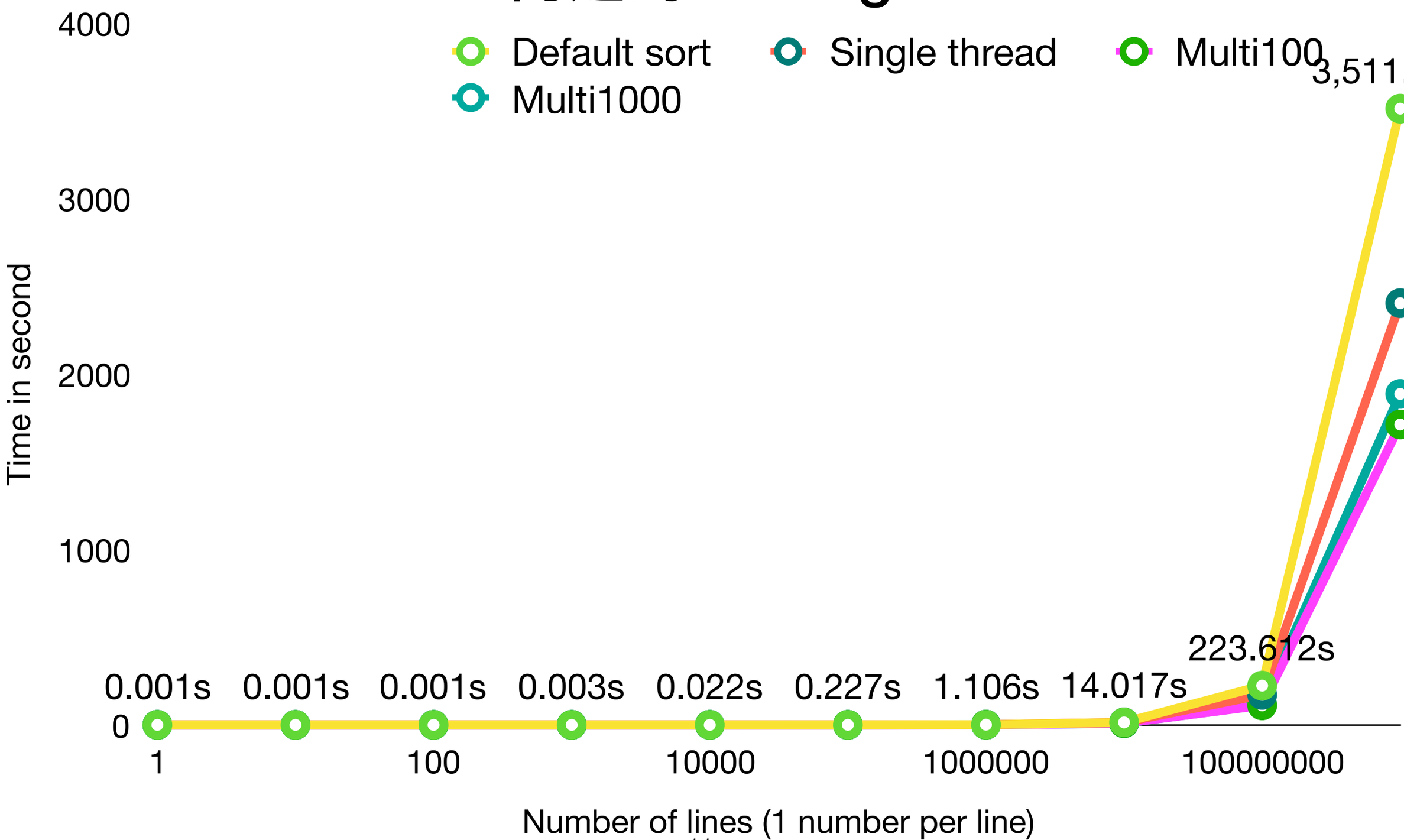
小結

- 工學院電腦 ram 太大，內建 sorting 會比較快一些
    - 我的winner tree貌似有寫爛，跟黃鈺程的比較，performance 大概會慢 20%

效能

linux.cs.ccu.edu.tw

內建的 sorting



效能

linux.cs.ccu.edu.tw

單線程sorting

效能

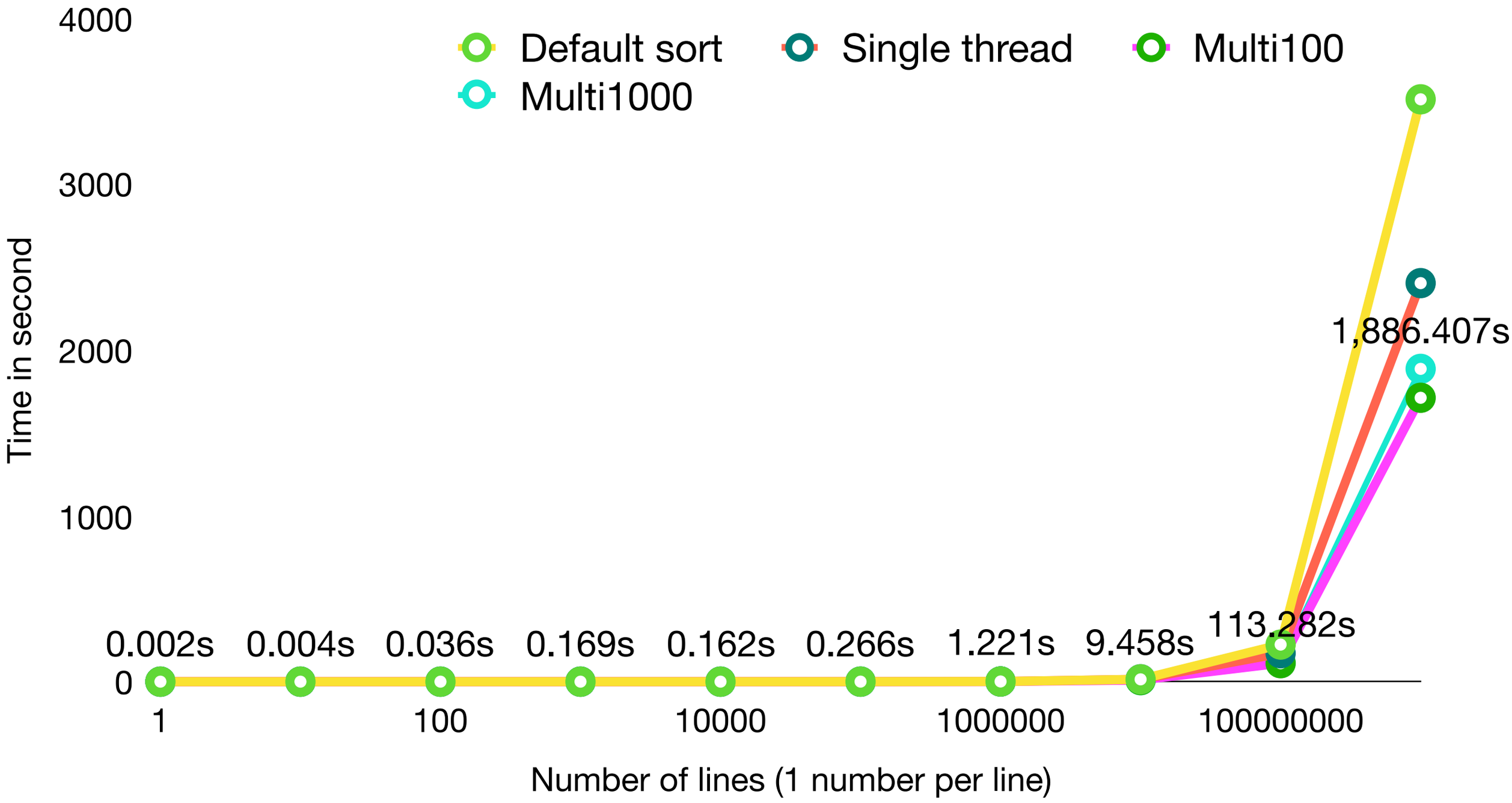
linux.cs.ccu.edu.tw

多線程, 100 runs, depth = 6

效能

linux.cs.ccu.edu.tw

多線程, 1000 runs, depth = 6



小結

- Linux 工作站電腦 ram 不大，內建 sorting 會比較慢一些

- Chunk原則上越少越快，但是受限於記憶體至少也要開個 50 chunks 才不會爆 ram