**2021 NYCU OS HW3 report**

|  |  |
| --- | --- |
| Question | Answer |
| Q1.  Briefly describe your design for the sorting algorithm, merge function, the thread management.  Also, describing the number of sort threads and number of merge threads in the Multi-thread program. | 我把partition\_num、thread\_num用define定義，之後要改的時候改一個地方就好比較方便。  我的partition\_num要是2的次方。  1.  把陣列切成partition\_num個，切的方式是紀錄每個partition的起始位置和結束位置，每個partition彼此互斥。  2.  因為partition合併後會需要繼續合併，所以也把合併後的位置紀錄，一直紀錄直到只剩一個partition。  3.  對每個partition做bubble sort，這邊可以用multithread加速。  4.  把每個sort 好的partition 兩兩merge起來，一直重複merge直到剩下一個partition，這邊也可以用multithread加速。  multithread加速的方法：把thread要做的那個function的參數填在二維陣列中，根據thread num決定填在二維陣列的哪一個row，再建立thread\_num個thread，一次傳一個row的參數進去，在main thread中join這thread\_num個thread。  我本來是在for填好一個thread要做的那個function的參數就建立一次thread，但好像會因為main thread繼續執行而把參數覆寫，create出的thread拿到的參數有時候就會出錯，所以我才換成填在二維陣列中。 |
| Q2.  Show the fastest time acceleration between single-thread and multi-thread. (Take screenshots of the time between single-thread and multi-thread) | Multithread: 4 threads, 8 partitions    Single thread: 1 thread, 8 partitions    437/112 = 3.9，將近4倍的加速，與預期滿接近的。 |
| Q3.  You need a brief description of the best  multi-threads and worst multi-threads methods.  The content includes the number of threads used and the way of partitioning, comparing the difference in time, and taking the screenshot between two multi-thread results. | Multithread best: 4 threads, 8 partitions    Multithread worst: 4 threads, 4 partitions    228/117=1.948，在這個case，2倍的partition有將近2倍的加速。  我partition的方式一樣，只是切的數量不一樣而已，程式碼也只有改最前面define partition\_num的值而已。 |
| Q4.  What did you learn from doing hw3? | 學到了如何使用thread和mutex lock，本來只有上課知道概念而已，實際操作後就更熟悉了。  且體會到main thread和create出來的thread可能會因為context switch而有非預期的結果，且這種bug不是每次都會出現，所以讓我非常難debug。  過程中也為了不要讓他因為context switch而有問題所以用到mutex lock，但有個地方感覺不該有問題卻一直找不到錯，所以最後改用二維陣列放參數，也就不需要mutex lock了。  所以這次讓我真正意識到multithread在寫的時候真的要非常小心，不然會debug到天荒地老都找不出問題。 |