標題：HW4-report
姓名：葉怡君
學號：110062529

Implementation

- Jobtracker(master)：

先將 input file 一行行讀進來，也會順便依據給定的 chunk size 進行合併，接著就讀取 locality table，等 tasktracker 來要任務，當有 tasktracker 傳 nodeid 過來 request 時，就依據下圖的 algo 尋找對應的 task 並傳送 taskid 給那個 tasktracker，若 mapper task 分配完會傳送-1 給 slave 以告知。

Algo:
1. Receive requests from nodes.
2. Return the first task with data locality to the requesting node.
3. If no task with locality, return the first available task.

接著就等著 slave 傳值回來告知它們已將此 task 執行完成，若收到的值為 taskid 則會再收到一個執行秒數，若收到-1 代表 tasktracker 的所有 task 都執行完成。

接著就開始分配 reduce task，分配方法和得知 task 是否完成皆如上述。

- Tasktracker(slave)：

在跟 jobtracker 要工作的時候會先傳送自己的 nodeid 過去，若還有 mapper task 沒被分配完，就會 recv 到 taskid，若 mapper task 都分配完就會收到-1。在得知自己要處理的全部 task 後，就幫每個 task(chunk) create mapper thread，若分配到的 task 數量多於 cpu 數量，就會先把一些已經 create 過的 thread 先 join 起來，再 create 新的 thread。

每個 mapper thread 都會依序執行 split function、map function、partition、跟 local write。

- Split function：將一個 line split 成多個 words 回傳
- Map function：將每個 word map 成(‘word’ ,1)
- Partition function：將每個 key 根據它的第一個 character mod (reduce task 的數量)為這個 key-value pair 分配到的 reduceid
- local write：根據分配到的 reducerid 寫入各自的 file

每一個 map task 執行完後，都會傳 taskid 跟所花費的時間給 jobtracker 知道，傳-1 時代表 map task 都執行完成。

接著用一樣的方法取得 reduce task，create thread 的方式：因為 reducer thread 只有一個，因此 create 完就會馬上 join，每一個 reduce task 執行完後，都會傳 reduceid 跟所花費的時間給 jobtracker 知道，傳-1 時代表 map task 都執行完成。

每個 reducer thread 都會依序執行 sort function、group function、reduce function、跟 output。

- Sort function：依字母順序排序，可選擇 ascending or descending
- Group function：default 把同樣的 key group 起來，格式如 →
  (‘key’ ,[1,1,1,1])

- Reduce function：將 value 加總 → ('key' ,4)

Challenge

- 在記錄 log 的部分，由於我的 sort(shuffling)包含在每個 reduce task 裡，因此不太能 output 出 shuffling time。
- 如何讓 jobtracker 跟 tasktracker 互相得知彼此的狀態，只能靠 MPI_Send 跟 MPI_Recv 互相傳送訊息，但一個不小心，可能某一方沒有要 send 訊息了，但另一方卻持續監聽著，就有可能卡在那裏。
- 在 output 的部分，我看到 sample output 中每個檔案的行數都很平均，且字母的排序是有順著接下來的，對我來說這個蠻難的，也是我沒有做到的地方。
- 整體來說，我覺得要充分理解 mapreduce 的流程&架構，才可以知道哪部分要用 process、哪部分要用 thread，我在理解的部分花了蠻長的時間，但也不確定我寫的架構是否正確。

Conclusion

因為自己這學期有修巨量資料這門課，課程中也有提到 mapreduce，因此有種熟悉感，但是當時的作業只教我們如何"使用"那些 function，在這裡實作 mapreduce，讓我更了解它底層的架構以及複雜之處，而且充分的理解為什麼會有 pyspark 的存在，因為 key-value pair 的組成不一定是 string 跟 int，可以是各種 type，我在這次作業中，也定義蠻多種 type 的。

```
typedef pair<string, int> Item;
typedef pair<int, string> Item2;
typedef pair<string,vector<int>> Item3;
ofstream logg;

typedef struct map_arg{
    int chunkid;
    string s;
    int reducer;
}MAPARG;

typedef struct map_arg2{
    int redu_taskid;
    string name;
    string dir;
}MAPARG2;
```