MDA_HW1 report

此次作業主要是參考 WordCount 這個官方範例的架構去做修改，並且使用 1 STEP 的 MapReduce 實作 Martix 相乘 M*N，另外再 main function 中可以調整矩陣大小，分別為”m” “j” “k”，因為這次 input 是 500X500，所以 m j k 預設都為 500，再由 configuration 在其他 function 中取得。

Mapper：

收先取得 M 矩陣的 rows 的數量(input file 的大小是 500)，以及 N 矩陣的 cols 的數量(input file 的大小是 500)，用 Configuration 這個 Hadoop 提供的 object 去儲存(code 裡儲存了 M 的 rows　N 的 rows cols，因為 M 的 cols 數一定和 N 的 rows 數相同，所以只需用一個變數儲存)，使用 String 得 split 將一行 text 作分割得到(mapAndreduce)，並且能從第一個字串去判斷是矩陣 M 還是 N，接著 M matrix 照著書上公式((i,k)(‘M’,j,m_ij))的 key value pair 寫入，用一個 for loop 跑到 N 的 cols 數量，由上述公式分別填入

Key(mapAndreduce[1]+","+Integer.toString(i)))
Value(mapAndreduce[0]+","+mapAndreduce[2]+","+mapAndreduce[3])

而 N marix 的公式則是((i,k),(‘N’,j,n_jk))的 key value pair 寫入，用一個 for loop 跑到 M 的 rows 數量，由上述公式分別填入

Key(Integer.toString(i)+","+mapAndreduce[2]))

Value(mapAndreduce[0]+","+mapAndreduce[1]+","+mapAndreduce[3])

如此一來就完成了 Map 的部分

Reducer：

由於這裡不須寫出 Key 的值，所以我在 extend class 時，第三個參數就給了 NullWritable，否則你在 context.write 時，如果填入 null 會發生 error。我在這裡創了兩個 HashMap m 與 n(方便操作 有大量的 build in function 以及尋找資料的 time complexity 小於 Linked list)用來儲存傳入的 value 中的相互對應，剛剛的 value 中有三個數值，分別代表，該 pair 屬於哪一個 matrix，以及用來辨認相乘的代號也就是 m_ij*n_jk 中的 j，最後一個是該點的數值，如果是屬於 M 矩陣，存入 m 中，否則則存入 n 中。

接著處理M-row 與 N-col相乘的運算，這裡需要取得N的rows數量，一樣用Configuration取得，使用for loop跑到N的rows數量，使用HashMap方便的function，先判斷有沒有存在用containsKey()，再來用get()取得該點的數值，result = result + m_ij*n_jk;，直到跑出for loop就能得到key 這個位置上的value