Readme of Assignment #5

1. Word Count

Basic idea: first using report.getInputSplit to get the name of the current processing file, and map the name to reduce procedure with keys of “word”. Then, due to the fact that the values passed from the map procedure would be sorted, we can define a variable called “lastfilename” to save the filename that the reduce procedure is going through, then, the key with multiple same filename would not count if the present filename equals to the filename that stored in that variable. And only different file name would increase the counter, then stores the different filename into “lastfilename”, and then following repeated filename would not increase the counter. After finishing reducing, Then we set the counter to reduced value. No combiner is used, and it may could enlarge the communication loads.

2. Matrix Multiplication

Basic idea: the program has three inputs, the matrix, the vector and the output path. The path of the vector should put into the distributed cache files. And get the vector in configure() before map procedure starts. What should be noticed is that the vector is indexed beginning with 0 whereas the matrix is index beginning with 1. so column index should minus 1 when multiplying each elements of the vector. No combiner is used.

3. Jacobi Calculation

Basic idea: the program has 3 inputs, matrixAb, max iterations, and significant digits. In main function, we can get the counter from the running job, and see whether it equals to 0 (which means that all the elements – Xval from the last iteration would meet the requirement of significant digits), if it equals, then we break the loop. In each loop we

restart a new job and read the cache file for both Matrix Ab due to we should set Aii, Ain arrays for each iteration and Xval in last iteration. And the program should get them in configure() in both map&reduce procedures.

What should be noticed is that, due to the Matrix is n*(n+1), the element whose column equals to n should not be mapped to reduce procedure; Moreover, due to the program would open a new job for each iteration, the counter, and the parameters such like matrix size, eps, and so on, should be passed to configure() for each time. Each iteration would cost nearly 5000~6000ms.

As for parallelism, the program has test for different number of map tasks, which are 1,2 ,4,8 (in crowded environment with several jobs running, test matrix: .4dat). And it seems that maps' time cost are basically same. It is possible that each map task should take times to prepare some initialization works, (function configure()) and their cost may cover the benefits of parallisms.