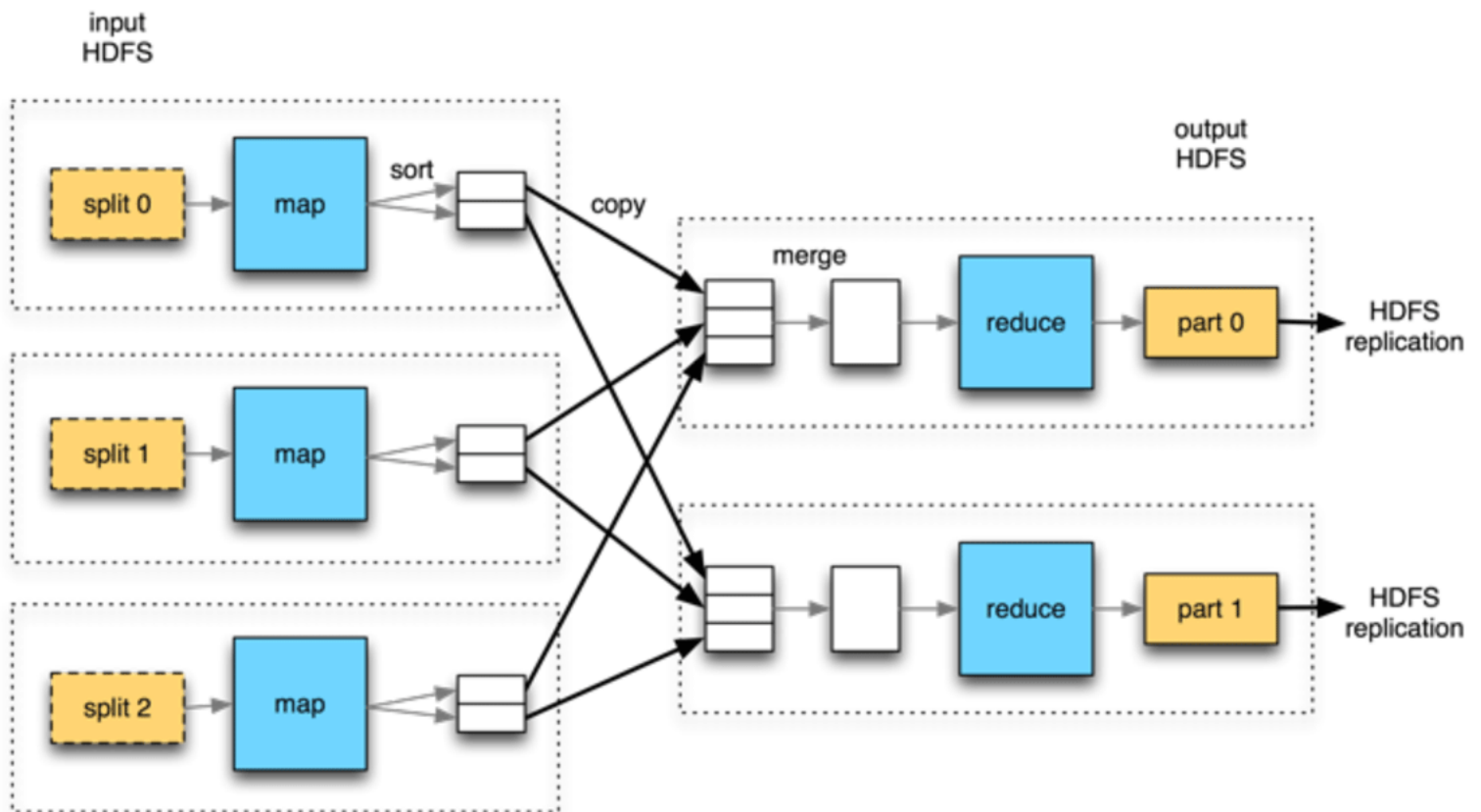


# Project 3: Report

## CSCI: 493.66 Map Reduce Using Hadoop

Akanksha Verma - May 14, 2014



<http://twimags.com/ddj/images/article/2013/0313/hadoopfig2.gif>

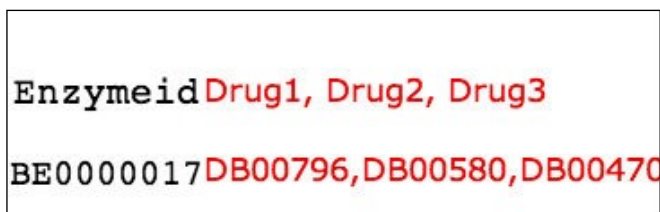
---

## Algorithm Design

Starting out with this project, with basic understanding of Map() and Reduce() functions in Hadoop , applying the brute force approach was to group the drugs the the common enzymes. Grouping the drug Ids for a common enzyme would automatically give us a group from which the pairs of drugs that react with common enzymes can categorized.

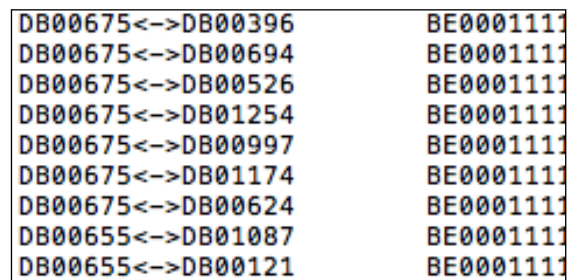
After further researching the functions, in the first step the Enzyme id was selected as the key and drug id was the value against it. The mapper function, went through each line in the input file ( with the format (Drugid : Enzymeid) ),then mapped it into ( EnzymeId

Drugid).This step still generated repeats of Enzyme - Drug pairing. Here each drug id



```
Enzymeid Drug1, Drug2, Drug3
BE0000017DB00796,DB00580,DB00470
```

Fig 1



```
DB00675<->DB00396      BE0001111
DB00675<->DB00694      BE0001111
DB00675<->DB00526      BE0001111
DB00675<->DB01254      BE0001111
DB00675<->DB00997      BE0001111
DB00675<->DB01174      BE0001111
DB00675<->DB00624      BE0001111
DB00655<->DB01087      BE0001111
DB00655<->DB00121      BE0001111
```

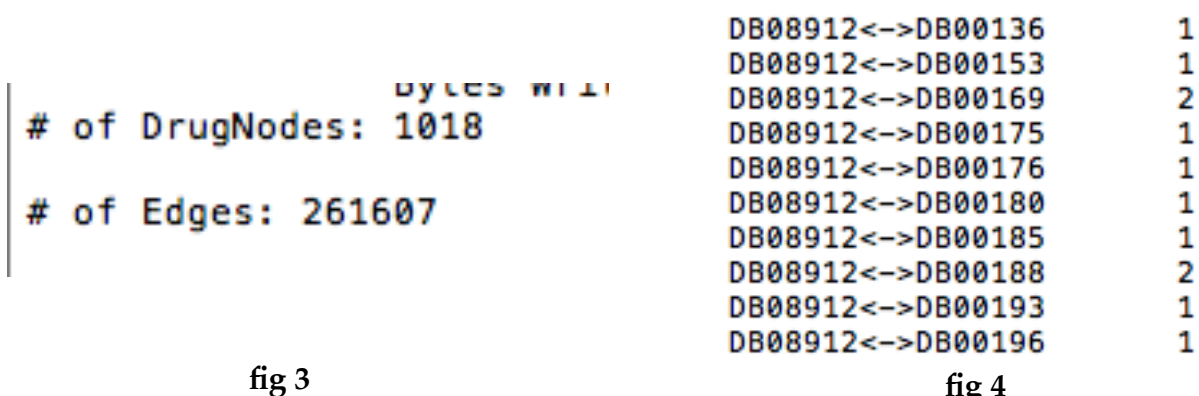
Fig 2

being read were entered into a data structure such as a hash set in order to store each unique Drug id found. The size of this hash set later give us the number of unique drug nodes present in relational data. Therefore in order to avoid redundancy and to move to the next step of the algorithm, the reducer function is called. The reducer function, goes through the new output from the Mapper function and removes repeated Enzyme id(key). As it removes repeated Enzyme id, it takes the Drugid corresponding to a particular Enzyme Id and appends it into a long string of Drug Id as value(see fig 1). Once the group of drugs common to an enzyme was appended to a string, the next step was to permute combination of the druids in order to form pairs. The string of drug ids is first split based on the delimiter and tokenized into individual tokens. Then using double while loop, the tokens are combined to form pairs by generating (n choose k ) combinations. where n would be the number of druids in the group and k =2 . Once the reducer function, the output is reformatted to appear as seen in Fig 2, with the drug id pairs now formed. Also, now the reducer, maps the Drugid pairs as Keys and the corresponding common enzyme as values. As seen in fig 1, this generated data shows redundancy with the enzyme ids being repeated.

I tried to not use another map reduce function and reduce and calculate common enzyme in this reduce function itself, but it took another set of Map reduce. The second map reduce was used to work

on output generated from first set of map reduce(fig1). After the second Map-reduce functions, they data was further sorted, reduced and recombined to a more precise list with unique Drug-Drug pairs and the number of common enzymes they shared. The number of common enzymes they shared were aggregated by taking the drug-drug pair as the key. Finally, counting the drug drug pair also have us the number of edges present in the relational graph between the Drug ids that share common enzymes.

The final output ,we got was , as shown in the (fig4) on the right. With drug id pairs as keys , and the number of common enzymes as values. The output is generated as an executable in the output folder passed in the command line. At the end from the graphing calculations, the number of unique Drug Nodes and the Number of edges were also obtained. These were outputted on the console itself as shown in fig(3).



Sorting and shuffling being inbuilt features of the map reduce functions were extremely helpful and helped avoid extra implementation of sort functions. They sorted the keys in order , giving back a very organized list of output. The java implementation was then made using the apache.hadoop libraries with hadoop2.2.0 eclipse plugin, for a single system implementation.

*\*More information on how to run the program and other files may be found in the README.txt file*