CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

This thesis has presented a web conscious distributed indexing algorithm based on HSim Framework which has been implemented on MapReduce Model. LSI combined with K-means has proved to be a scalable information retrieval for the web conscious. The investigational results shown that the scalability has been achieved to some extent. However, a new challenge has been introduced, which is the overhead of the clustering algorithm k-means. The investigational results have indicated that when the size of the document corpus increases to a certain level, the overhead of the pre-clustering becomes highly considerable due to both the computing complexity and the limitations of the computing resources. MapReduce-LSI successfully solves such an issue based on distributed k-means algorithm in a Hadoop cluster. The experimental results have shown that with a proper number of centroids, the recall and precision of the MR-LSI algorithm become highly close to the typical standard LSI. In terms of algorithm executing time, due to the system overhead of the Hadoop framework, it is quite high with processing small size of document collections. The work also has shown, that when the size of the document collection increases to a certain critical boundary, the overhead of the framework can be overcome, which means that MR-LSI outperforms the other algorithms in terms of overhead with maintaining the similar recall and precision levels.
The MR-LSI algorithm has been evaluated in a Hadoop cluster which contains a number of four nodes. The study is extendable to the performance of MR-LSI with mass computing resources. To study the scalability of the MR-LSI algorithm, a simulator has been developed to perform a number of simulations. The simulator proposed in the thesis HSim modeled the parameters of Hadoop framework from several aspects including node parameters, cluster parameters and Hadoop system clusters. These parameters can help HSim to create a simulated Hadoop cluster with detailed specifications which are mainly employed in a real cluster. The validations have shown that the performance of HSim is quite close to that of the practical experimental cluster. And also HSim outperforms MRPerf of which the performance is highly different from that of experimental environment.

Using HSim, a simulated Hadoop cluster with the number of 25 nodes up to 250 nodes has been created. Thus from 100 to 1000 mappers are involved to evaluate the scalability of MR-LSI. The evaluations have indicated that generally along with the number of mappers increased, the performance of MR-LSI got enhanced in terms of overhead. However, due to the wave mechanism in the Hadoop framework, at the points of certain number of mappers, simply keeping increasing the number of mappers cannot achieve the performance enhancement. Adequate numbers of tests have also been done to study the impacts brought by tuning parameters on the MR-LSI algorithm. The results have indicated that the performance of the algorithm can be significantly affected by the different configurations of the cluster.

Only two types of the strategies FIFO and Fair Scheduler have been supported until now. The two types of the schedulers aim to balance the resources among jobs of different users. However, as the framework supports heterogeneous nodes, only balancing the resources among users may not get the satisfactory optimized performance due to the unbalanced load among mappers employed by different nodes. Thus a static load balancing strategy has
been proposed by this work. In the strategy, the working mechanism of mapper is modeled from four aspects which include copying time, processing time, spilling time, and merging time. The copying time represents the time of copying a data chunk to local hard disk of the mappers. The processing time represents the actual data processing by a processor. The spilling time represents the time of emptying a buffer while the buffer is filled up. The merging time represents the time of merging intermediate data into a whole chunk which will be ultimately sent to reducer(s). By modeling these overheads of working mechanism of mappers, the data sizes that are initially sent to mappers involved in the processing can be calculated. Therefore to balance the load among mappers, according to a certain scheduler, if the overall overhead which is the sum of the above four sub-overheads of each mapper could be close enough to those of the other mappers. And then the scheduler can be regarded as the best solution.

Instead of stiffly and directly measuring a solution with a complex way, the Mean Square Error (MSE) has been used. MSE can represent how different a series of data is. Therefore, by calculating the MSE of all mappers' overhead, the differences among them can be quantitatively measured. Aiming to get the optimized solutions from the combinations of a set of complex equations, the genetic algorithm has been involved. The genes are the volume of data to be allocated to mappers while the chromosomes are the schedulers and the fitness uses MSE. Within a number of generations, an optimized scheduler can be found. In a static environment, as long as the scheduler has been worked out, the mappers can use the scheduler in the whole data processing duration until the job is finished. The evaluations of the load balancing algorithm have been done in a simulated cluster with heterogeneous. The concept of heterogeneity is involved to measure the level of differences among nodes employed in the cluster. The evaluated results show that:
The load balancing algorithm considerably enhances the performance of the cluster when the heterogeneity increases to a certain level. When the levels of heterogeneity are lower, due to the overhead of the load balancing algorithm itself, it cannot outperform the scheduler without load balancing in terms of overhead. However, with large levels of heterogeneity, the algorithm can be three times faster compared to the scheduler without load balancing strategy.

The load balancing algorithm is suitable for jobs with different sizes. In the simulation result, it indicates that the when size of data increases from 10GB to 100GB, the algorithm can give stable performance with varying sizes of data.

Though along with the increasing number of mappers and size of data, the overhead of the load balancing algorithm keeps increasing, compared with the enhancement gained by the algorithm, the impact brought by the overhead is negligible.

However, frequently a cluster is not simply static but dynamic. There are lots of factors affecting the computing capacity of a cluster dynamically along with the time passes. To balance the load among mappers in a dynamic computing environment, a dynamic load balancing strategy has been proposed by this work. The strategy consisted of two components.

1. For each processing wave the data selection solution has been given to decide the volume of data. The objective is trying to use the higher computing capacity time interval to process the data. The algorithm will be executed in next wave again to correct the error caused by the IO operations. The data selection also results in dynamic window sizes in launching the load balancing algorithm.
2. In the dynamic environment the copying time, processing time, spilling time, and merging time have been modeled. Based on the equations of these four times, the overall overhead of a mapper can be represented. To conclude the relationships between sizes of data and the allocated mappers can be established.

Owing to the complexity of the equations or the optimized scheduling, genetic algorithm has been involved which the fitness function is based on calculating the MSE of the overhead of mappers. A number of researchers claimed that with a less number of generations the GA can gain the optimized solution. However, based on the experimental results, it is not suitable for the Hadoop framework. Therefore, to reduce the overhead of the genetic algorithm, combining with the characteristics of the Hadoop framework, an improvement has been done for the genetic algorithm, which can significantly reduce the number of generations. The simulator Hsim also offers a way based on two features to create a dynamic environment. The dynamic features include speed of hard disk and load of processor. The dynamic factor of hard disk can create dynamic IO environments for the cluster. Similarly the dynamic factor of the processor load can create dynamic computing capacity of the cluster. This work presents two different kinds of processor loads of which one is simple and the other one is complex. Compared with the performances of computing ratio strategy and fixed window size strategy, the dynamic load balancing algorithm achieves significant enhancement when the level of heterogeneity is larger than a certain value. Thus, based on different heterogeneities, a number of evaluations have been done.
6.2 FUTURE ENHANCEMENT

The work presented in this thesis unlocks a new way to build up a web Conscious distributed Indexing algorithm for scalable information retrieval. However the experimental and simulation result of the algorithm demonstrates a reasonable performance, it is obvious that several challenges exist and some of them are listed below:

1. In the future, this MapReduce framework can be extended using different optimization algorithms for better results. The execution time can be further reduced to excel the proposed method.

2. Finding the best value of K disjoint subsets and N_j data points to minimize the sum of square criterion so as to achieve the global minimum can be considered in future.

3. The time interval of executing load balancing GA further reduced to achieve the overhead by controlling iterations.

4. Interactions among HDD, CPU & Load Generator further improved by using a better mathematical model. So as to gain high accuracy in HSim.

5. Determining best latent chromosome to mutate with several genes in the chromosome to avoid the local optimum of the algorithm, based on transformation probability.

6. The performance of MR-LSI can be enhanced further by finding some other combination portioning algorithm with LSI.

7. The proposed architecture can be evolved further to address the larger cluster of web conscious information ecosphere.