Zhoujian Lan 27827903
Mingzhou Lin 40047923

SON Algorithm based on MapReduce Introduction

Introduction of MapReduce

Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically, both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks. The MapReduce framework operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The SON Algorithm and MapReduce

The SON algorithm lends itself well to a parallel-computing environment. Each of the chunks can be processed in parallel, and the frequent itemsets from each chunk combined to form the candidates. We can distribute the candidates to many processors, have each processor count the support for each candidate in a subset of the baskets, and finally sum those supports to get the support for each candidate itemset in the whole dataset. This process does not have to be implemented in

MapReduce, but there is a natural way of expressing each of the two passes as a MapReduce operation. We shall summarize this MapReduce-MapReduce sequence below.

First Map Function: Take the assigned subset of the baskets and find the itemsets frequent in the subset using the algorithm of A-Priori Algorithm. The output is a set of key-value pairs (F, 1), where F is a frequent itemset from the sample. The value is always 1 and is irrelevant.

First Reduce Function: Each Reduce task is assigned a set of keys, which are itemsets. The value is ignored, and the Reduce task simply produces those keys (itemsets) that appear one or more times. Thus, the output of the first Reduce function is the candidate itemsets.

Second Map Function: The Map tasks for the second Map function take all the output from the first Reduce Function (the candidate itemsets) and a portion of the input data file. Each Map task counts the number of occurrences of each of the candidate itemsets among the baskets in the portion of the dataset that it was assigned. The output is a set of key-value pairs (C, v), where C is one of the candidate sets and v is the support for that itemset among the baskets that were input to this Map task.

Second Reduce Function: The Reduce tasks take the itemsets they are given as keys and sum the associated values. The result is the total support for each of the itemsets that the Reduce task was assigned to handle. Those itemsets whose sum of values is at least s are frequent in the whole dataset, so the Reduce task outputs these itemsets with their counts. Itemsets that do not have total support at least s are not transmitted to the output of the Reduce task.

In addition, in order to implement a frequent Item Set Mining Algorithm using

Map-Reduce by restricting the main memory of r M, the large data sets are split before the first MapReduce process.

Optimization

At the beginning of the project, we try to apply one MapReduce phase to deal with the whole data sets. However, we realize that this thought are not able to handle large data sets, like 1000 baskets.

After searching several papers, we decide to use two MapReduce phases. We discover that SON algorithm is fit to MapReduce. For this reason, we decide to use SON based on MapReduce. In this way, we can split a large data set into several small data sets and process them separately.

Contribution

Each of the member contributes more or less equally in the project in term of coding, research, and discussion.

In early stage, every member contributes same on searching appropriate algorithms which can solve this problem. After trying two algorithms, we dicide to use SON-mapreduce algorithm.

In middle stage, we divided the program into two parts: the first phase of map-reduce and the second phase of map-reduce. Mingzhou Lin completed the first phase of map-reduce(Include apriori algorithm), Zhoujian Lan finished the second phase of map-reduce.

In final stage, we write this report and complete the test to our program together.