FiDoop-DP: Data Partitioning in Frequent Itemset Mining on Hadoop Clusters

OBJECTIVE:

The objective of this system is to achieve compressed storage and avoid building conditional pattern bases, FiDoop incorporates the frequent items ultra metric tree, rather than conventional FP trees.

ABSTRACT:

Traditional parallel algorithms for mining frequent itemsets aim to balance load by equally partitioning data among a group of computing nodes. We start this study by discovering a serious performance problem of the existing parallel Frequent Itemset Mining algorithms. Given a large dataset, data partitioning strategies in the existing solutions suffer high communication and mining overhead induced by redundant transactions transmitted among computing nodes. We address this problem by developing a data partitioning approach called FiDoop-DP using the MapReduce programming model. The overarching goal of FiDoop-DP is to boost the performance of parallel Frequent Itemset Mining on Hadoop clusters. At the heart of FiDoop-DP is the Voronoi diagram-based data partitioning technique, which exploits correlations among transactions. Incorporating the similarity metric and the Locality-Sensitive Hashing technique, FiDoop-DP places highly similar transactions into a data partition to improve locality without creating an excessive number of redundant transactions.

INTRODUCTION:

Frequent itemsets mining (FIM) is a core problem in association rule mining (ARM), sequence mining, and the like. Speeding up the process of FIM is critical and indispensable, because FIM consumption accounts for a significant portion of mining time due to its high computation and input/output (I/O) intensity. When datasets in modern data mining applications become excessively large, sequential FIM algorithms running on a single machine suffer from performance deterioration. To address this issue, we investigate how to perform FIM using Map Reduce—a widely adopted programming model for processing big datasets by exploiting the parallelism among computing nodes of a cluster. We show how to distribute a large dataset over the cluster to balance load across all cluster nodes, thereby optimizing the performance of parallel FIM.