FiDoop: Parallel Mining of Frequent Item sets Using Map Reduce

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:

Existing parallel mining algorithms for frequent item sets lack a mechanism that enables automatic parallelization, load balancing, data distribution, and fault tolerance on large clusters. As a solution to this problem, we design a parallel frequent item sets mining algorithm called FiDoop using the Map Reduce programming model. To achieve compressed storage and avoid building conditional pattern bases, FiDoop incorporates the frequent items ultra metric tree, rather than conventional FP trees. In FiDoop, three Map Reduce jobs are implemented to complete the mining task. In the crucial third Map Reduce job, the mappers independently decompose item sets, the reducers perform combination operations by constructing small ultra metric trees, and the actual mining of these trees separately. We implement FiDoop on real cloud storage. We show that FiDoop on the cluster is

sensitive to data distribution and dimensions, because item sets with different lengths have different decomposition and construction costs. To improve FiDoop's performance, we develop a workload balance metric to measure load balance across the cluster's computing nodes. We develop FiDoop-HD, an extension of FiDoop, to speed up the mining performance for high-dimensional data analysis. Extensive experiments using real-world celestial spectral data demonstrate that our proposed solution is efficient and scalable.

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.