www.ijiarec.com

ISSN:2348-2079

Volume-8 Issue-3

International Journal of Intellectual Advancements and Research in Engineering Computations

TRIDENT: Distributed storage, analysis, and exploration of multidimensional phenomena

K.Saranya[1], C.Saranya[2]

[1]Assistant Professor, Department of Computer Science and Engineering, Bharathiyar Institute of Engineering for Women, Deviyakurichi, Attur- 636112

[2]PG Scholar Master of Engineering, Department of Computer Science and Engineering, Bharathiyar Institute of Engineering for Women, Deviyakurichi, Attur- 636112

ABSTRACT

Rising storage and computational capacities have led to the accumulation of voluminous datasets. These datasets contain insights that describe natural phenomena, usage patterns, trends, and other aspects of complex, real-world systems. Statistical and machine learning models are often employed to identify these patterns or attributes of interest. However, a wide array of potentially relevant models and parameterizations exist, and may provide the best performance only after preprocessing steps have been carried out. Our distributed analytics platform, TRIDENT, facilitates the modeling process by providing high-level data exploration functionality as well as guidance for creation of effective models. TRIDENT handles [1] data partitioning and storage, [2] metadata extraction and indexing, and [3] selective retrievals or transformations to prepare and generate training data. In this study, we evaluate TRIDENT in the context of a 1.1 peat byte epidemiology dataset generated by a disease spread simulation; such datasets are often used in planning for national-scale outbreaks in animal populations.

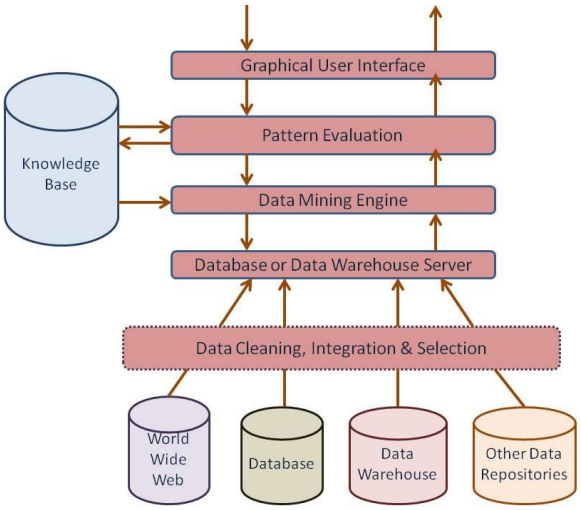
Keywords: High-Level Data, Voluminous Data Management, Data Partitioning and Storage, Effective Models.

INTRODUCTION

We are in an age often referred to as the information age. In this information age, because we believe that information leads to power and success, and thanks to sophisticated technologies such as computers, satellites, etc., we have been collecting tremendous amounts of information. Initially, with the advent of computers and means for mass digital storage, we started collecting and storing all sorts of data, counting on the power of computers to help sort through this amalgam of information. Unfortunately, these massive collections of data stored on disparate structures very rapidly became over whelming. This initial chaos has led to the creation of structured databases and database management systems (DBMS). The efficient database management systems have been very important assets for management of a large corpus of data and especially for effective and efficient retrieval of particular information from a large collection whenever needed. The proliferation of database management systems has also contributed to recent massive gathering of all sorts of information. Today, we have far more information than we can handle: from business transactions and scientific data, to satellite pictures, text reports and military intelligence. Information retrieval is simply not enough anymore for decision-making. Confronted with huge collections of data, we have now created new needs to help us make better managerial choices. These needs are automatic summarization of data, extraction of the "essence" of information stored, and the discovery of patterns in raw data [4-7].

Author for correspondence:
Department of Computer Science and Engineering, Bharathiyar Institute of Engineering for Women, Deviyakurichi, Attur- 636112

Data Mining Architecture



RELATED WORK

The graphs used in Galileo share some common features with k-d trees, but do not employ binary splitting and allow much greater fan-out as a result. Similar to Tries, identical attributes in a record can be expressed as single vertices, which simplify traversals and can reduce memory consumption. However, Galileo graphs support multiple concurrent data types, maintain an explicit feature hierarchy (that can also be reoriented at runtime), and employ dynamic quantization through configurable tick marks. Mongo DB shares several design goals with Galileo, but is a document-centric storage platform that does not support analytics directly. However, Mongo DB has rich geospatial indexing capabilities and supports dynamic schemas through its JSON-inspired binary storage format, BSON. Mongo DB can use the Geohash algorithm for its spatial indexing functionality, and is backed by a B-tree data structure for fast lookup operations. For load balancing and scalability, the system supports sharding ranges of data across available computing and storage resources, but imposes some limitations on the breadth of analysis that can be performed on extremely large datasets in a clustered setting [8-10].

Facebook's Cassandra project is a distributed hash table that supports column-based, multidimensional storage in a tabular format. Like Galileo, Cassandra allows user-defined partitioning schemes, but they directly affect lookup operations as well; for instance, using the random data partitioner backed by a simple hash algorithm does not allow for range queries or adaptive changes to the partitioning algorithm at runtime. This ensures that retrieval operations are efficient, but also limits the flexibility of partitioning schemes. Cassandra scales out linearly as more hardware is added, and supports distributed computation through the Hadoop runtime. Predictive and approximate data structures are not maintained by the system itself, but could be provided through additional preprocessing as new data points are added to the system.

LITERATURE SURVEY

Matthew Malensek, SangmiPallickara and ShrideepPallickara

As remote sensing equipment and networked observational devices continue to proliferate, their corresponding data volumes have surpassed the storage and processing capabilities of commodity computing hardware. This trend has led to the development of distributed storage frameworks that incrementally scale out by assimilating resources as necessary. While challenging in its own right, storing and managing voluminous datasets is only the precursor to a broader field of research: extracting insights, relationships, and models from the underlying datasets. The focus of

this study is twofold: exploratory and predictive analytics over voluminous, multidimensional datasets in a distributed environment. Both of these types of analysis represent a higher-level abstraction over standard query semantics; rather than indexing every discrete value for subsequent retrieval, our framework autonomously learns the relationships and interactions between dimensions in the dataset and makes the information readily available to users.

Konstantin Shvachko; HairongKuang ;

Sanjay Radia

The Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers both host directly attached storage and execute user application tasks. By distributing storage and computation across many servers, the resource can grow with demand while remaining economical at every size. We describe the architecture of HDFS and report on experience using HDFS to manage 25 pet bytes of enterprise data at Yahoo!

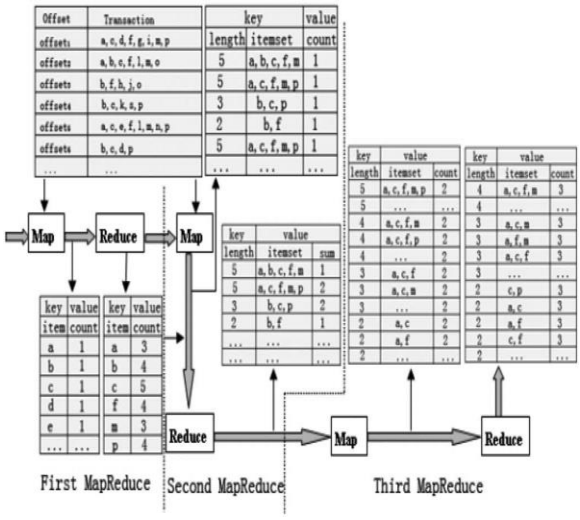
PROPOSED SYSTEM

In Proposed System a new data partitioning method to well balance computing load among the cluster nodes; we develop FiDoop-HD, an extension of FiDoop, to meet the needs of high-dimensional data processing.

Advantages

- FiDoop is efficient and scalable on Hadoop clusters.
- Reducing I/O overhead,
- Offering a natural way of partitioning a dataset,
- Compressed storage, and
- Averting recursively travers

SYSTEM MODEL



MODULES

- Frequent Itemset Mining
- MapReduce Framework
- Parallel FP-Growth Algorithm

Frequent itemset mining

Frequent Itemset Mining is one of the most critical and time-consuming tasks in association rule mining (ARM), an often-used data mining task, provides a strategic resource for decision support by extracting the most important frequent patterns that simultaneously occur in a large transaction database. A typical application of ARM is the famous market basket analysis.

Map reduce framework

MapReduce is a popular data processing paradigm for efficient and fault tolerant workload distribution in large clusters. A MapReduce computation has two phases, namely, the Map phase and the Reduce phase. The Map phase splits an input data into a large number of fragments, which are evenly distributed to Map tasks across a cluster of nodes to process. Each Map task takes in a key-value pair and then generates a set of intermediate key-value pairs. After the MapReduce runtime system groups and sorts all the intermediate values associated with the same intermediate key, the runtime system delivers the intermediate values to Reduce tasks

Parallel Fp-Growth algorithm

This is based on a popular FP-Growth algorithm called Parallel FP-Growth or Pfp for short. Pfp implemented in Mahout is a parallel version of the FPGrowth algorithm. Mahout is an open source machine learning library developed on Hadoop clusters. FP-Growth efficiently discovers frequent itemsets by constructing and mining a compressed data structure (i.e., FP-tree) rather than an entire database. Pfp was designed to address the synchronization issues by partitioning transaction database into independent partitions, because it is guaranteed that each partition contains all the data relevant to the features (or items) of that group.

METHODOLOGY

Frequent items ultra-metric trees method

We made a complete overhaul to FIUT (i.e., the frequent items ultra-metric trees method), and addressed the performance issues of parallelizing FIUT. We developed the parallel frequent itemsets mining method (i.e., FiDoop) using the MapReduce programming model. We proposed a data distribution scheme to balance load among computing nodes in a cluster. We further optimized the performance of FiDoop and reduced running time of processing high-dimensional datasets. We conducted extensive experiments using a wide range of synthetic and real-world datasets, and we show that FiDoop is efficient and scalable on Hadoop clusters. After the root is labeled as null, an itemset p1, p2, . . . , pm of frequent items is inserted as a path connected by edges (p1, p2), (p2, p3), . . . , (pm−1, pm) without repeating nodes, beginning with child p1 of the root and ending with leaf pm in the tree. An FIU-tree is constructed by inserting all itemsets as its paths; each itemset contains the same number of frequent items. Thus, all of the FIU-tree leaves are identical height. Each leaf in the FIU-tree is composed of two fields: named item-name and count. The count of an item-name is the number of transactions containing the itemset that is the sequence in a path ending with the item name. Non leaf nodes in the FIU-tree contain two fields: named item-name and node-link. A node-link is a pointer linking to child nodes in the FIU-tree.

CONCLUSION

To solve the scalability and load balancing challenges in the existing parallel mining algorithms for frequent item sets, we applied the MapReduce programming model to develop a parallel frequent item sets mining algorithm called FiDoop. FiDoop incorporates the frequent items ultra-metric tree or FIU-tree rather than conventional FP trees, thereby achieving compressed storage and avoiding the necessity to build conditional pattern bases. FiDoop seamlessly integrates three MapReduce jobs to accomplish parallel mining of frequent item sets. The third MapReduce job plays an important role in parallel mining; its mappers independently decompose item sets whereas its reducers construct small ultra-metric trees to be separately mined. We improve the performance of FiDoop by balancing I/O load across data nodes of a cluster.

REFERENCES

[1]. M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, ser. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, 10–10.

[2]. C. Lam, Hadoop in Action. Greenwich, CT, USA: Manning Publications Co., 2010.

[3]. M. Abadi, P. Barham, J. Chen, "Tensorflow: A system for large-scale machine learning," in Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, ser. OSDI'16. Berkeley, CA, USA: USENIX Association, 2016, 265–283. http: //dl.acm.org/citation.cfm?id=3026877.3026899

[4]. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," J. Mach. Learn. Res., 12, 2011, 2825–2830.

[5]. K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), ser. MSST '10. Washington, DC, USA: IEEE Computer Society, 2010, 1–10. Available: http://dx.doi.org/10.1109/MSST.2010.5496972

[6]. M. Zaharia, "Resilient distributed datasets: A faulttolerant abstraction for in-memory cluster computing," in Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, 2–2.
Available: http: //dl.acm.org/citation.cfm?id=2228298.2228301

[7]. C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," ACM Trans. Intell. Syst. Technol., 2(3), 2011, 1–27. Available: http: //doi.acm.org/10.1145/1961189.1961199

[8]. R Core Team, "R: A language and environment for statistical computing," https://www.r-project.org/.

[9]. Pandas Developers, "Pandas python data analysis library," http: //pandas.pydata.org.

[10]. N. Harvey, A. Reeves, M. Schoenbaum., "The North American Animal Disease Spread Model: A simulation model to assist decision making in evaluating animal disease incursions," Preventive Veterinary Medicine, 82(3), 2007, 176–197.