Trends in the Solution of Distributed Data Placement Problem

Rohini T V
Asst. Prof, Dept. of ISE,SJBIT, Bengaluru, India
Research Scholar, VTU, Belagavi
e-mail: rohinitv@gmail.com

Ramakrishna M V
Professor, Dept. of ISE
SJBIT, Bengaluru, India
e-mail: mvrama@yahoo.com

Abstract**—** Data placement for optimal performance is an old problem. For example the problem dealt with the placement of relational data in distributed databases, to achieve optimal query processing time. Heterogeneous distributed systems with commodity processors evolved in response to requirement of storage and processing capacity of enormous scale. Reliability and availability are accomplished by appropriate level of data replication, and efficiency is achieved by suitable placement and processing techniques. Where to place which data, how many copies to keep, how to propagate updates so as to maximize the reliability, availability and performance are the issues addressed. In addition to processing costs, the network parameters of bandwidth limitation, speed and reliability have to be considered. This paper surveys the state of the art of published literature on these topics. We are confident that the placement problem will continue to be a research problem in the future also, with the parameters changing. Such situations will arise for example with the advance of mobile smart phones both in terms of the capability and applications.

Keywords- *Data placement; Replication; Distributed system*

*****

I. INTRODUCTION

In the modern computing environment processors, memory and internet have become ubiquitous. Companies such as yahoo, ebay, facebook, google are dealing with the problem of processing enormous amount of data which is being produced continuously. For example Google processed 20 petabytes of data per day as of 2009 [5, 16]. Similar volumes of data are being dealt with by other companies. The main issue here is the amount of data and the speed with which data need be processed for user requests. These companies resorted to non-traditional approach of horizontal scaling where a large number of commodity hardware is used to store and process the data. Google developed Google File System (GFS), to store their data on a large number of processors, each of them being a commodity machine [6]. To deal with semi-structured data they developed Bigtable data model and processing system [3]. Google file system is an example of a distributed file system which enables storing and processing of large volumes of data on thousands of commodity processors. In such a system in general, all the processors may not be identical in terms of the processing capacity as well as the storage capacity. When a query comes to a machine to be processed, it would access the data stored on various other machines. The access delay will vary depending on how data is placed or distributed on the machines. Accessing data on the disk of the same machine is fastest. Next is accessing from the same rack followed by machines located in the same data center which have to go through network switches, and lastly from remote location. As the time progresses the query pattern changes; in other words the access pattern and nature of queries will vary and not stay static. Hence the proper placement of data which enables high availability with minimum delay is important.

The goal of solution to data placement problem is to achieve best throughput, minimize delay and maximize the utilization of resources. The problem of data placement has manifested in various forms for a long time in computer science. This paper deals with the research which has addressed the problem for the newly emerged heterogeneous environment as discussed above. In the next section we describe the Google file system, the most significant distributed file system today. This is followed by QFS (Quantcast file system) in the section 3 which deals with wide area placement of data replicas, in which the main aim is to maintain multiple copies at multiple locations. We follow this with a discussion of automatic and location aware data placement techniques in section 4.

II. GOOGLE FILE SYSTEMS

Google File System(GFS) is designed to provide efficient, reliable access to data using large clusters of commodity hardware [6]. GFS provides a familiar file system interface. Files are organized hierarchically in directories and identified by pathnames. GFS supports the usual operations to create, delete, open, close, read, and write files and GFS has snapshot and record append operations. Snapshot creates a copy of a file or a directory tree at low cost. Record append allows multiple clients to append data to the same file concurrently while guaranteeing the atomicity of each individual clients append. A GFS cluster consists of multiple nodes. These nodes are divided into two types: one Master node and a large number of Chunkservers. Files are divided into fixed-size chunks. Chunkservers store these chunks. Each chunk is assigned a unique 64-bit label by the master node at the time of creation, and logical mappings of files to constituent chunks are maintained. Each chunk is replicated several times throughout the network, with the minimum being three, but even more for files that have high end-in demand or need more redundancy.

The Master server doesn’t usually store the actual chunks, but rather all the metadata associated with the chunks, such as the namespace, access control information, the mapping from files to chunks, and the current locations of chunks, what processes are reading or writing to a particular chunk, or taking a ”snapshot” of the chunk pursuant to replicate it (usually at the instigation of the Master server, when, due to node failures, the number of copies of a chunk has fallen beneath the set number). All this metadata is kept current by the Master server periodically receiving updates from each chunk server ”Heart-beat” messages. GFS client code linked into each application implements the file system API and communicates with the master and chunk servers to read or write data on behalf of the application. Clients or applications interact with the master for metadata operations, but all data-bearing communication goes directly to the chunkservers. GFS typically has hundreds of chunkservers spread across many machine racks. These chunkservers in turn may be accessed from hundreds of clients from the same or different racks Communication between two machines on different racks may cross one or more network switches. When the user population is scattered over the entire world, communicating with every user machine over a short network path is impossible. Large-scale Internet services often solve this problem by means of replication. For reliability, each chunk is replicated on multiple chunkservers The chunk replica placement policy serves two purposes maximize data reliability and availability, and maximize network bandwidth utilization. The master re-replicates a chunks soon as the number of available replicas falls below a user-specified goal. This could happen for various reasons: a chunkserver becomes unavailable; it reports that its replica may be corrupted, one of its disks is disabled because of errors, or the replication goal is increased. Finally, the master re-balances replicas periodically. It examines the current replica distribution and moves replicas for better disks pace and load balancing.

III. Wide-area Placement of Data Replicas

In a distributed system in general data is often replicated at multiple locations for the sake of reliability, and kept consistent to serve user requests [18, 7]. The replica placement serves two purposes: maximize data reliability and availability and maximize network band width. Ping et. al. address the wide area placement of data replicas for fast and highly available data access[12]. They have addressed the problem of how to place replicas in wide area networks, where data is replicated at multiple locations to serve users with lower latency and higher availability. Having replicas in wide area applications leads to fast and highly available data access [8]. Since it is in-feasible to know in advance the user access pattern of data, this research focuses on determining replica locations based on the past data accesses. A main challenge that arises in this case is to analyze data access patterns in an efficient, scalable manner so as to deal with a large number of users. Such analysis of data accesses needs to be done efficiently even across geographic regions since each data replica can be accessed by a different user population. The analysis must facilitate robust estimation of both data access delay and availability for feasible replica placement scenarios [17]. Here optimal replica locations are chosen, further, redirecting the applications requests to a nearby replica. In this work they proposed two algorithms, one for finding the optimal replica locations and another to create user cluster node .

The first algorithm returns highest utility value and replica locations that lead to value. The algorithm chooses set of replica locations that lead to optimal solution from the candidate locations. In this algorithm, let L be the candidate locations, T is the set of already chosen replica locations and main aim is to maximize the objective function (delay, availability). In this approach, data access pattern is collected by forming cluster of user nodes. For each replica placement plan R, the objective function needs to be evaluated based on the estimated average data access delay and availability. It would be impractical to collect information about all the users across data replicas and store it at a central server. Thus, this approach maintains for each replica, a summary of the coordinates of the users that have recently accessed the replica.

Whenever r new locations of replicas need to be determined, these summaries are collected at a central location and then used for estimating the overall data access delay and availability for each replica placement under consideration. The second algorithm describes the manner in which the coordinates of users are summarized at each replica location and the coordinate of such users are classified int m clusters. The distance between user nodes and replica locations is obtained from network coordinate systems RNP [13]. By knowing coordinates of replica locations, it can predict the nearest replica to access data with high accuracy.

Assaf and Danny deal with replica placement strategy in which a replica must be synchronized with the original content server in order to supply reliable and precise service to the client requests [2]. Replica placement across data centers is a very common approach for improving performance and availability of content services. Content replication algorithms deploy a set of servers, distributed throughout the data centers network, and replicate the relevant data across these servers. Both the time required to access the data and the traffic in the network are reduced by Replica placement deals with the actual number and network location of the replicas. Clearly, would like to minimize the network distance between an email application and the closest replica containing the desired content (in this example the authentication server) and thus having more replicas helps On the other hand, having more replicas is more expensive so it needs to model the cost and the benefit in a way that can allow to make the appropriate decisions regarding the number and the network locations of the replicas. This problem is strongly related to a family of optimization problems generally referred to as facility location problems [15, 1]. Most of the algorithms neglect the cost of keeping the replicas across the network up to date. In this work, A replica must be synchronized with the original content server in order to supply reliable and precise service to the client requests. The amount of synchronization traffic across the network depends on the number of replicas deployed in the network, the topology of the distributed update and the rate of updates in the content of the server.

Chervenak and Deelman deal with Data Placement For Scientific Applications In Distributed Environments [4]. The aim of this work is to distribute data and to make it advantages for application execution. They study the relationship between data placement services and work flow management systems and placement activity data sets are largely asynchronous with respect to work flow execution. TevFik and Miron focus on the idea of placement of data in distributed computing systems similar to the I/O subsystem in operating system [9]. In this data placement strategy different data transfer protocols may have different optimum concurrency level for any two source and destination. They have not focused on optimal location with respect user access pattern..

IV. Automatic Data placement

The problem of automatic data placement with replicated key value stores is address by Joao et. al. [11]. The main objective is to automatically optimize the replica placement in a way that locality patterns in data accesses, such that the communication between nodes is minimized. The issues addressed are, the placement of the objects generating most remote operation for each node and combining the usage of consistent hashing with al data structure.

Yu and pan addressed the problem of placement in three scenarios. The first case without replicas, the problem is addresses with hyper-graph formulation. In the second scenario with replicas, the numbers of replicas allowed are considered. In the third scenario replica migration is considered In this paper [19] data placement problem solved using three scenarios, first, for the scenario without replicas, in this scenario placement problem is solved by the hyper-graph formulation. In the second scenario, with replicas and the number of replicas allowed are considered. In the third scenario replica migration is considered. Amol et. al. [14] deals with a technique to address the data placement problem called SWORD: a scalable workload-aware data partitioning and placement approach. In which the problem addressed in OLTP systems. The techniques introduced by them are to significantly reduce the overheads of initial placement of data and also during processing of the query. They have used hyper-graph compression technique over the data items. Further they address data repartitioning technique which modifies data placement without complete workload repartitioning. The technique enables availability and load balancing. The availability-aware data placement problem to improve the application performance without Extra storage cost has been addressed by Yang et. al. [7]. They proposed a technique of ADAPT which deals with Hadoop framework and the performance of ADAPT is evaluated in non-dedicated distributed environment. ADAPT is to migrate data based on the availability of each node, reduce network traffic, improve data locality, and optimize the application performance.

V. CONCLUSIONS: FUTURE OF PLACEMENT PROBLEM

Although data placement problem is old (such as in relation to OLTP systems), the engineering details have changed considerably in the present computing environment. We have massively parallel processing and storage capability with commodity hardware, that is distributed widely in geographic terms. The parameters of the problem include the network bandwidth, limitations of maximum capability, as well as delays. In this paper we have provided a survey of the research literature on the solution techniques and issues addressed. We discussed the balanced data placement problem and its solution for distributed system. We see this problem will continue to be significant research issue in foreseeable future with new environments emerging. The smart phone getting smarter and more powerful will be a source of more information all over the world, and accordingly new engineering details will emerge for the data placement problem. New research will have to address these issues.

References

[1] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. SIAM Journal on Computing, 33(3):544–562, 2004.

[2] Danny Raz Assaf Rappaport. Update aware replica placement. In Proceedings of the 9th

[3] CNSM and Workshops, pages 92–99. IFIP, 2013.

[4] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable: A distributed

[5] Ann Chervenak, Ewa Deelman, Miron Livny, Mei-Hui Su, Rob Schuler, Shishir Bharathi, Gaurang Mehta, and Karan Vahi. Data placement for scientific applications in distributed environments. In Proceedings of the 8th IEEE/ACM International Conference on Grid Computing, GRID '07, pages 267–274, Washington, DC, USA, 2007. IEEE Computer Society.

[6] E. F. Codd. A relational model of data for large shared data banks. Commun. ACM 13(6):377–387, June 1970.

[7] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In ACM SIGOPS Operating Systems Review, pages 29–43. ACM, 2003.

[8] Hui Jin, Xi Yang, Xian-He Sun, and Ioan Raicu. Adapt: Availability-aware mapreduce data placement for non-dedicated distributed computing. In Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on, pages 516–525. IEEE, 2012.

[9] Magnus Karlsson and Christos Karamanolis. Choosing replica placement heuristics for wide-area systems. In Distributed Computing Systems, 2004. Proceedings. 24th International Conference on, pages 350–359. IEEE, 2004.

[10] Tevfik Kosar and Miron Livny. A framework for reliable and efficient data placement in distributed computing systems. Journal of Parallel and Distributed Computing 65(10):1146–1157, 2005.

[11] Michael Ovsiannikov, Silvius Rus, Damian Reeves, Paul Sutter, Sriram Rao, and Jim Kelly. The quantcast file system. Proceedings of the VLDB Endowment, 6(11):1092– 1101, 2013.

[12] Joao Paiva, Pedro Ruivo, Paolo Romano, and Lu´ıs Rodrigues. A uto p lacer: Scalable selftuning data placement in distributed key-value stores. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 9(4):19, 2015.

[13] Fan Ping, Jeong-Hyon Hwang, XiaoHu Li, Chris McConnell, and Rohini Vabbalareddy. Wide area placement of data replicas for fast and highly available data access. In Proceedings of the fourth international workshop on Data-intensive distributed computing, page1–8. ACM, 2011

[14] Fan Ping, Christopher McConnell, and Jeong-Hyon Hwang. A retrospective approach for accurate network latency prediction. In Computer Communications and Networks (ICCCN), 2010

Proceedings of 19th International Conference on, pages 1–6. IEEE, 2010.

[15] Abdul Quamar, K Ashwin Kumar, and Amol Deshpande. Sword: scalable workload-aware data placement for transactional workloads. In Proceedings of the 16th International Conference on Extending Database Technology, pages 430–441. ACM, 2013.

[16] David B Shmoys, Eva Tardos, and Karen Aardal. Approximation algorithms for facility location problems. In Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, pages 265–274. ACM, 1997

[17] Michael Stonebraker. Sql databases v. nosql databases. Commun. ACM, 53(4):10–11, April 2010.

[18] Michal Szymaniak, Guillaume Pierre, and Maarten Van Steen. Latency-driven replica placement. In Applications and the Internet 2005.Proceedings. The 2005 Symposium on, pages 399–405. IEEE, 2005.

[19] Radu Tudoran, Alexandru Costan, Rui Wang, Luc Boug´e, Gabriel Antoniu, et al. Bridging data in the clouds: An environment-aware system for geographically distributed datatransfers. In 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing,2014 .

[20] Boyang Yu and Jianping Pan. Location-aware associated data placement for geo-distributed data-intensive applications. In 2015 IEEE Conference on Computer Communications (INFOCOM), pages 603–611. IEEE, 2015.