International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.5, Sep-Oct. 2012 pp-3743-3746 ISSN: 2249-6645

Reducing Real Time Service Delays Using Map reduce Frame Work

K.M.DhanaSailaja[1], E.V.Prasad[2]
(M.Tech, University College of Engineering JNTUK, Andhra Pradesh, India)
(Professor & Registrar, University College of Engineering JNTUK, Andhra Pradesh, India)

ABSTRACT: *Recently the demand of real-time data services has increased in various applications such as manufacturing, Web-servers, and e-commerce and they are becoming sophisticated in their real-time data needs. Real time data services has to provide better QoS parameters because end users may ignore the service when the database service delay is high and temporally inconsistent data is available. Due to dynamic workloads providing better QoS in data services is a challenging issue. In this paper to overcome this problem we are using map-reduce framework to estimate the real time service delays. MapReduce framework process large amount of data in a parallel way. It is gaining lot of interest in data mining, because the programmer is abstracted to the data storage, distribution, replication, load balancing and it uses functional programming. It has two functions map and reduce. Several experiments have been conducted on various data sets to calculate the performance of the proposed technique.*

Keywords **:** Web-servers, Map Reduce, e-commerce, Data mining, Quality of service

I. INTRODUCTION:

Recently the demand of real-time data services has increased in various applications such as manufacturing, Web-servers, and e-commerce and they are becoming sophisticated in their real-time data needs [1], [2]. In the past decade demand is increasing provisioning for quality of service (QoS) guarantees to various network applications and clients. The data normally span from low-level control data, typically acquired from sensors, to high-level management and business data. In these applications, it is desirable to process user requests within the timeline using fresh data. In dynamic systems, such as Web servers and sensor networks with non-uniform access patterns, the workload of real-time databases (RTDB) cannot be predicted accurately. Hence, the RTDBs can be overloaded, uncontrolled deadline misses and data freshness may not be possible during the transient overloads. To provide service quality we propose a quality of service (QoS) sensitive method that provides a set of requirements to improve the performance of the database even in the unpredictable workloads conditions. In some applications like Web service it is desirable that the QoS does not vary significantly from one transaction to another. It is emphasized that individual QoS needs requested by transactions are enforced and any deviations from the QoS needs should be uniformly distributed among the clients to ensure QoS fairness. Imprecise computation techniques [3] have been proposed by various authors to allow flexibility and for achieving graceful degradation during transient overloads. These techniques make it possible to trade off resource needs for the quality of a requested service and they have been successfully applied in timeliness is emphasized applications where a certain degree of imprecision can be tolerated [4], [5], [6]. Real time data services has to provide better QoS parameters because end users may ignore the service when the database service delay is high and temporally inconsistent data is available. Due to dynamic workloads providing better QoS in data services is a challenging issue. In this paper to overcome this problem we are using map-reduce framework to estimate the real time service delays. Our MapReduce framework process large amount of data in a parallel way and it is gaining lot of interest in data mining, because the programmer is abstracted to the data storage, distribution, replication, load balancing and it uses functional programming. It has two functions map and reduce. In the map stage passes the data over the input file and outputs (key, value) pairs, the shuffling stage transfers the mappers output to the reducers based on the key and finally the reducer processes the received pairs and outputs the final result. Map Reduce is a useful tool for large amount of data analysis because of its scalability, simplicity and low cost to build large clouds of computers.

The rest of the paper is organized as section 2: discuss about the related work, section 3: presents the Proposed Solution, section 4: discuss about Map Reduce, section 5: discuss about Experimental setup, section 6: concludes the paper.

II. RELATED WORK:

In e-commerce, the real time data service quality is mainly determined by both networks and data transfer rate between disks. The quality of service is mainly depending on the whole data that contain embedded objects. All the existing techniques, measure service quality with respect to a single packet in networks [11], [13], [15] or an individual request [7], [9], [12], [17], [21] or connection [20], [22], [24] in Web servers. C. Dovrolis [15], proposed a technique to provide different QoS levels between multiple aggregated traffic classes within a network. T.F. Abdelzaher [7], bound the server-side delay of individual client request. This algorithm mainly focuses on providing differentiated services to different client classes using priority-based scheduling [14], [16]. The aim of the algorithm is to provide better services to the premium class than to the basic class by adjusting the priority of the allocated processes between the classes on either user level or kernel level. Existing techniques are not providing any guarantee to QoS. To overcome this problem authors proposed queuing-theoretic approaches. It is well known that the delay upper bound in a G/G/1 is determined by the

system load and the variance of user's requests inter arrival and service time distributions. B. Urgaonkar [24] proposed an approach in which the resource allocation is controlled to adjust the load of a class so that the delay equals the upper bound and the performance of the approach is highly depend on the variance estimation, which is very difficult to measure accurately in the dynamic workload cases. Traditional linear feedback control is used to control the resource allocation in Web servers [7], [21]. Because the behavior of a Web server changes continuously, the performance of the linear feedback control is limited.

To overcome the problem of lack of accurate server model A. Kamra and M. Karlsson [17, 18] the parameters of the controllers were adjusted based on the workload. V. Sundaram [23] presented an approach where the resource allocation was controlled based on the past allocations of delivered service quality. All these approaches provide better performance than nonadaptive linear feedback control approaches under dynamic workload situations. Later B. Li [19] presented a fuzzy control model to address the nonlinear QoS requirements of different multimedia applications under different resource constraints. In [11], author proposed a set of rules to dynamically adjust the target delay ratios between various traffic flows to reduce the effect of bursty traffic in the basic class on the delay of the premium class.

III. PROPOSED SOLUTION:

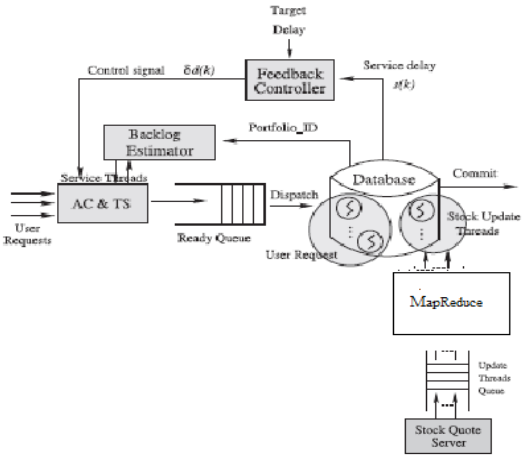

Fig 1: Proposed architecture

Figure 1 shows the proposed architecture of the database system. In this architecture we added the Map Reduce to the Chronos architecture [29] to provide better QoS in real time data services. It has database backlog estimator, admission controller, traffic smoother, feedback controller, MapReduce and a database server block. The feedback controller is designed based on either linear control theory or fuzzy logic control. The database server processes service requests and updates stock prices periodically received from the stock quote server to provide the updated stock prices. We consider periodic temporal updates that are commonly used in RTDBs for data temporal consistency [30] and a fixed time interval is selected for updating each stock price in a range [0.2 s, 5 s].

A periodic temporal updates are rarely considered in RTDBs due to the difficulties for defining and maintaining the notion of temporal consistency, backlog estimation for periodic updates and estimating the database backlog to service user data service requests. The dedicated update threads are scheduled in a separate queue ahead of user requests and they have to wait for next available threads. High priority is assigned to temporal data updates to provide data freshness. Map Reduce process large amount of data in a parallel way and it has two functions map and reduce. In the map stage passes the data over the input file and outputs (key, value) pairs, the shuffling stage transfers the mappers output to the reducers based on the key and finally the reducer processes the received pairs and outputs the final result.

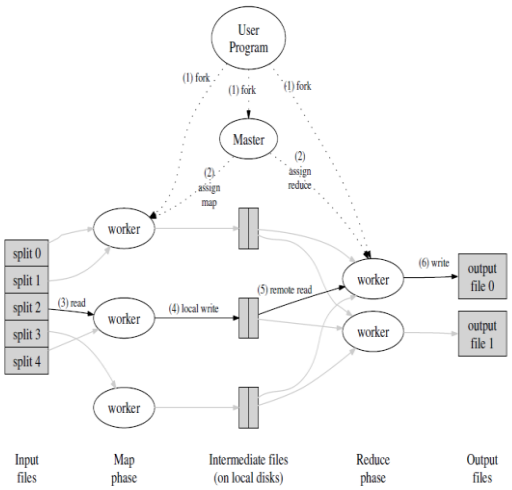

Fig 2: Block diagram of MapReduce

IV. MAPREDUCE:

Map Reduce framework process large amount of data in a parallel way and it is gaining lot of interest in data mining, because the programmer is abstracted to the data storage, distribution, replication, load balancing and it uses functional programming. It has two functions map and reduce. In the map stage passes the data over the input file and outputs (key, value) pairs, the shuffling stage transfers the mappers output to the reducers based on the key and finally the reducer processes the received pairs and outputs the final result. Map Reduce is a useful tool for large amount of data analysis because of its scalability, simplicity and low cost to build large clouds of computers.

1.1. Programming Model

The Map Reduce takes a set of input key/value pairs and produces a set of output key/value pairs. It has two functions Map and Reduce. Map function written by the user takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups all intermediate values with same intermediate key and passes them to the Reduce function. Further Reduce function written by the user accepts an intermediate key and a set of values for that key. It merges together these values to form a possibly smaller set of values and zero or one output value is produced per Reduce invocation. Using an iterator the intermediate values are supplied to the reduce

function which allows to handle large values to fit into the memory.

1.2. Example

Consider the problem of counting the number of occurrences of each word in a large collection of documents. The user would write code similar to the following pseudo-code:

```
map(String key, String value):
// key: document name
// value: document contents
for each word k in value:
EmitIntermediate(k, "1");

reduce(String key, Iterator values):
// key: a word
// values: a list of counts
int res = 0;
for each j in values:
res += ParseInt(v);
Emit(AsString(res));
```

The map function ejects each word plus an associated count of occurrences. The reduce function adds all counts ejected for a particular word.

1.3. Types

The map and reduce functions called by the user have associated types:

map (k1,v1) = list(k2,v2)
reduce (k2,list(v2)) = list(v2)

The input keys and values are drawn from a different domain than the output keys and value and the intermediate keys and values are from the same domain as the output keys and values.

The main problem in MapReduce for clustering very large datasets is how to minimize the I/O cost and how to minimize the network cost among processing nodes. To overcome these problems we used the ParC (Parallel Clustering) and SnI (Sample-and-Ignore) method. The ParC [27] algorithm partition the input data and assign each partition to an individual system then each system group the data partition in to a cluster, named as β- clusters, and finally merge the β-clusters to get the final clusters. This algorithm reads the dataset once to minimize disk access which is the common way used by serial algorithms to decrease the computational costs. But it does not discuss how to minimize the network traffic [28]. To address this problem we used SnI method, it samples the input data and creates an initial set of clusters and later the input data is filtered to include unclassified elements. Finally the clusters found by the reducers are merged with the clusters from the sampling phase using the same merging strategies used in ParC.

V. EXPERIMENTAL SETUP:

To evaluate the performance of the proposed algorithm, we use three machines each of them is same configuration such as the dual core 1.6GHz CPU and 1GB memory with Linux operating system. A Chronos server, clients and a stock quote server run on each of them, respectively.

For 80% of time, a client issues a query about stock prices. In the remaining time client requests a portfolio browsing, purchase or sale transaction at a time. Most data service requests in e-commerce are queries. At the beginning of the experiment, the inter-request time (IRT) is randomly distributed in [3s, 4s]. At 200s, the range of the IRT is suddenly reduced to model heavy workload changes, and stays in the new range until the end of the experiment at 500s.

For performance comparisons, we consider three approaches shown in Table 1. Open is the basic Berkeley DB [26] without any control facility. AC model control the incoming transactions in proportion to the service delay error under overload. FC-Q finds the relation between the queue size and response time, similar to [25].

Table 1: Tested Approaches

OPEN	Pure Berkeley
AC	Ad-hoc admission control
FC-Q	Feedback control (FC)
MapReduce	MapReduce control

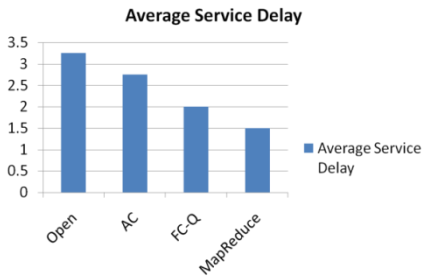

Fig 3: Average Service Delay

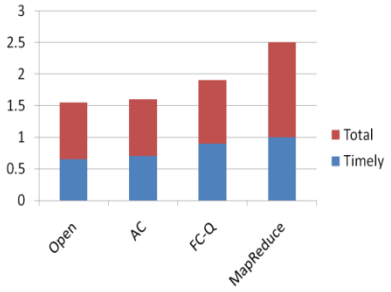

Fig 4: Number of Data processed

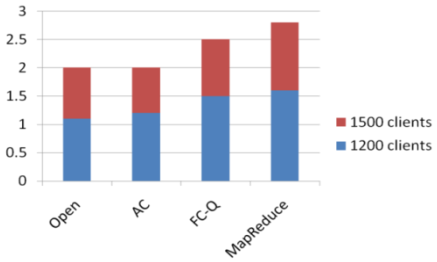

Fig 5: Number of Data processed

VI. CONCLUSION:

Real time data services has to provide better QoS parameters because end users may ignore the service when the database service delay is high and temporally inconsistent data is available. Due to dynamic workloads providing better QoS in data services is a challenging issue. In this paper to overcome this problem we are using map-reduce framework to estimate the real time service delays. MapReduce framework process large amount of data in a parallel way. It is gaining lot of interest in data mining, because the programmer is abstracted to the data storage, distribution, replication, load balancing and it uses functional programming. It has two functions map and reduce. Experimental results relived that our proposed approach support the desired average/transient data service delay and it provided better QoS parameters. It improved the service delay and throughput compared to other appraoches.

REFERENCES:

[1] D. Wu, Y.T. Hou, W.Z.Y.-Q. Zhang, and J.M. Peha, "Streaming Video over the Internet: Approaches and Directions," IEEE Trans. Circuits and Systems for Video Technology, vol. 11, no. 3, pp. 282-300, Mar. 2001.

[2] S.-Y. Choi and A.B. Whinston, "The Future of e-Commerce: Integrate and Customize," Computer, vol. 32, no. 1, pp. 133-134, Jan. 1999.

[3] J.W.S. Liu, W.-K. Shih, K.-J. Lin, R. Bettati, and J.-Y. Chung, "Imprecise Computations," Proc. IEEE, vol. 82, Jan. 1994.

[4] S. Zilberstein and S.J. Russell, "Optimal Composition of Real-Time Systems," Artificial Intelligence, vol. 82, nos. 1-2, pp. 181-213, 1996.

[5] X. Chen and A.M. K. Cheng, "An Imprecise Algorithm for Real-Time Compressed Image and Video Transmission," Proc. Int'l Conf. Computer Comm. and Networks (ICCCN), 1997.

[6] M. Yannakakis, "Perspectives on Database Theory," Proc. Ann. Symp. Foundations of Computer Science, 1995.

[7] T.F. Abdelzaher, K.G. Shin, and N. Bhatti, "Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach," IEEE Trans. Parallel and Distributed Systems, vol. 13, no. 1, pp. 80-96, Jan. 2002.

[8] M. Arlitt and T. Jin, "A Workload Characterization Study of the 1998 World Cup Web Site," IEEE Network, vol. 14, no. 3, pp. 30-37,May-June 2000.

[9] N. Bhatti and R. Friedrich, "Web Server Support for Tiered Services," IEEE Network, vol. 13, no. 5, pp. 64-71, 1999.

[10] P. Bhoj, S. Ramanathan, and S. Singhal, "Web2K: Bringing QoS to Web Servers," Technical Report HPL-2000-61, HP Laboratories,May 2000.

[11] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss,"An Architecture for Differentiated Services," IETF, RFC 2475, Dec. 1998.

[12] J.M. Blanquer, A. Batchelli, K. Schauser, and R. Wolski, "Quorum: Flexible Quality of Service for Internet Services," Proc. Symp. Networked Systems Design and Implementation, 2005.

[13] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: An Overview," RFC 1633, June 1994.

[14] J. Almeida, M. Dabu, A. Manikutty, and P. Cao, "Providing Differentiated Levels of Service in Web Content Hosting," Proc. ACM SIGMETRICS Workshop Internet Server Performance, pp. 91- 102, 1998.

[15] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling," IEEE/ACM Trans. Networking, vol. 10, no. 1, pp. 12-26, 2002.

[16] L. Eggert and J. Heidemann, "Application-Level Differentiated Services for Web Servers," World Wide Web J., vol. 2, no. 3, pp. 133-142, 1999.

[17] A. Kamra, V. Misra, and E. Nahum, "Yaksha: A Self Tuning Controller for Managing the Performance of 3-Tiered Websites,"Proc. Int'l Workshop Quality of Service, pp. 47-56, 2004.

[18] M. Karlsson, C. Karamanolis, and X. Zhu, "Triage: Performance Isolation and Differentiation for Storage Systems," Proc. Int'l Workshop Quality of Service, pp. 67-76, 2004.

[19] B. Li and K. Nahrstedt, "A Control-Based Middleware Framework for Quality of Service Adaptations," IEEE J. Selected Areas in Comm., vol. 17, no. 9, pp. 1632-1650, Sept. 1999.

[20] Y. Lu, T.F. Abdelzaher, C. Lu, L. Sha, and X. Liu, "Feedback Control with Queueing-Theoretic Prediction for Relative Delay Guarantees in Web Servers," Proc. IEEE Real-Time and Embedded Technology and Applications Symp., pp. 208-217, May 2003.

[21] P. Pradhan, R. Tewari, S. Sahu, A. Chandra, and P. Shenoy, "An Observation-Based Approach towards Self-Managing Web Servers," Proc. Int'l Workshop Quality of Service, 2002.

[22] L. Sha, X. Liu, Y. Lu, T.F. Abdelzaher, "Queueing Model Based Network Server Performance Control," Proc. IEEE Real-Time Systems Symp., pp. 81-90, 2002.

[23] V. Sundaram and P. Shenoy, "A Practical Learning-Based Approach for Dynamic Storage Bandwidth Allocation," Proc. Int'l Workshop Quality of Service, 2003.

[24] B. Urgaonkar, P. Shenoy, "Cataclysm: Handling Extreme Overloads in Internet Applications," Proc. Int'l World Wide Web Conf., May 2005.

[25] K.D. Kang, J. Oh, and S.H. Son, "Chronos: Feedback Control of a Real Database System Performance,"IEEE Real-TimeSystems Symp., 2007.

[26] "Oracle Berkeley DB Product Family," http://www.oracle.com/database/berkeleydb/index.html. 2010.

[27] Jianbin Wei, and Cheng-Zhong Xu, "eQoS: Provisioning of Client-Perceived End-to-End QoS Guarantees in Web Servers", IEEE Transactions On Computers, Vol. 55, No. 12, December 2006.

[28] Mehdi Amirijoo, Jorgen Hansson, and Sang Hyuk Son, "Specification and Management of QoS in Real-Time Databases Supporting Imprecise Computations", IEEE Transactions on Computers, Vol. 55, No. 3, March 2006.

[29] Kyoung-Don Kang, Yan Zhou, and Jisu Oh, "Estimating and Enhancing Real-Time Data Service Delays: Control-Theoretic Approaches" IEEE Transactions On Knowledge And Data Engineering, Vol. 23, No. 4, April 2011.