Disease-Treatment Relations Using Machine Learning Approach

Miss. Praveena. R[1], Prof. K. Chandra Prabha[2]
[1]*P.G. Student, Computer Application Department, Alagappa Chettiar Government College of engineering & Technology*
[2]*Head of the Department, Computer Application Department, Alagappa Chettiar Government College of engineering & Technology*

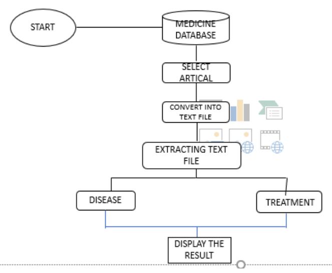
Abstract- **Based on the above analysis, it suffices to conclude that both N-way replication and erasure coding yield unbalanced tradeoff among read performance, write performance, and space efficiency. We argue and demonstrate through comprehensive experiments that, such imbalance causes two major problems. We design and implement disease, a hybrid storage scheme combining N-way replication and erasure coding, to provide high reliability, efficient I/O performance and flexible consistency simultaneously at low storage cost for object-based cloud storage systems. We propose MPL (Multi versioned Parity Logging) mechanism to facilitate efficient random write handling and read responding under both sequential and PRAM consistency. To verify the effectiveness of the proposed design, we evaluate disease on request-handling efficiency and the associated storage cost. We compare disease with 4-way CRAQ RS-code, two representative specific implementation of N-way replication and erasure coding. We set up wide range of system and workload configurations to investigate how the overall performance of a storage scheme is influenced when experiment parameters vary. We implement two consistency levels in the current disease system sequential consistency and PRAM (Pipeline RAM) consistency. These two consistency levels represent reverse consistency-performance tradeoff, and disease naturally supports both of them to meet different requirement in varied applications or services.**

Index Terms- **disease treatment, machine learning.**

I. INTRODUCTION

CLOUD storage system plays the role of storing and managing the data of upper-tier services and applications. From service provider's point of view, the need for high availability and performance is always placed at the forefront. High availability requires that the underlying storage system is equipped with certain level of fault-tolerance, and high performance demands efficient handling towards client-side read/write requests. Meanwhile, lower storage cost is always exhilarating for service providers. Storage cost includes the expenditure on storage space, network communication, and data transferring. The storage cost of a storage scheme is determined by the way it manages the data and the specific request handling protocols it establishes. Currently, two storage schemes are the most prevalently-applied for today's large-scale storage systems: N-way replication and erasure coding. N-way replication duplicates each piece of data into N replicas. These N replicas are stored on N geographically distributed servers, so the system can tolerate up to server-failure. Meanwhile, N-way replication guarantees low response latency and high throughput by spreading workload to different servers. Based on the above analysis, it suffices to conclude that both N-way replication and erasure coding yield unbalanced tradeoff among read performance, write performance, and space efficiency. We argue and demonstrate through comprehensive experiments that, such imbalance causes two major problems. To verify the effectiveness of the proposed design, we evaluate disease on request-handling efficiency and the associated storage cost. We compare disease with 4-way CRAQ RS-code, two representative specific implementation of N-way replication and erasure coding. We set up wide range of system and workload configurations to investigate how the overall performance of a storage scheme is influenced when experiment parameters vary.

II.SYSTEM ARCHITECTURE



Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There's no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate. The OLAP Services feature available in SQL Server version 7.0 is now called SQL Server 2000 Analysis Services. The term OLAP Services has been replaced with the term Analysis Services. Analysis Services also includes a new data mining component

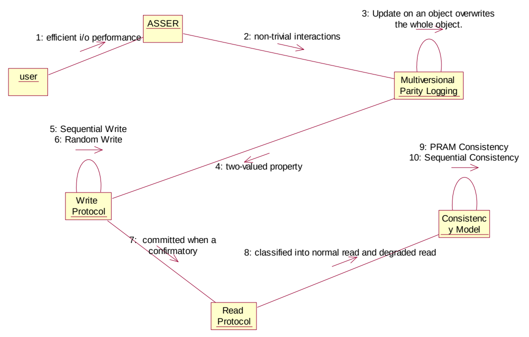
Constructors and destructors:
Constructors are used to initialize objects, whereas destructors are used to destroy them. In other words, destructors are used to release the resources allocated to the object. In C#.NET the sub finalize procedure is available. The sub finalize procedure is used to complete the tasks that must be performed when an object is destroyed. The sub finalize procedure is called automatically when an object is destroyed. In addition, the sub finalize procedure can be called only from the class it belongs to or from derived classes.

Garbage collection
Garbage Collection is another new feature in C#.NET. The .NET Framework monitors allocated resources, such as objects and variables. In addition, the .NET Framework automatically releases memory for reuse by destroying objects that are no longer in use. In C#.NET, the garbage collector checks for the objects that are not currently in use by applications. When the garbage collector comes across an object that is marked for garbage collection, it releases the memory occupied by the object.

Overloading
Overloading is another feature in C#. Overloading enables us to define multiple procedures with the same name, where each procedure has a different set of arguments. Besides using overloading for procedures, we can use it for constructors and properties in a class. A query is a question that has to be asked the data. Access gathers data that answers the question from one or more table. The data that make up the answer is either dynaset (if you edit it) or a snapshot (it cannot be edited).Each time we run query, we get latest information in the dynaset. Access either displays the dynaset or snapshot for us to view or perform an action on it, such as deleting or updating.

Multithreading:
C#.NET also supports multithreading. An application that supports multithreading can handle multiple tasks simultaneously, we can use multithreading to decrease the time taken by an application to respond to user interaction.

III. WORKING PROCEDURE


ASSER
user
Multiversional Parity Logging
Write Protocol
Read Protocol
Consistency Model
1: efficient i/o performance
2: non-trivial interactions
3: Update on an object overwrites the whole object.
4: two-valued property
5: Sequential Write
6: Random Write
7: committed when a confirmatory
8: classified into normal read and degraded read
9: PRAM Consistency
10: Sequential Consistency

Reliable
While a relatively uncomplicated design, ASSER introduces non-trivial interactions among components. Being stably efficient for diverse workloads requires storage system to deliver balanced I/O performance towards all types of client requests. To this end, we implement MPL (Multiversional Parity Logging) mechanism to

support the elementary read/write protocols of ASSER. MPL is an extended mechanism of prevalently-adopted Parity Logging technique. The benefits brought by MPL mechanism are two folded. Firstly, parity logging facilitates efficient handling towards update requests. The segment-chain in ASSER takes charge of receiving and handling update requests, thus reducing the amount of disk space needed to be overwritten.

Cost-effective
Each object can have more than one recoverable versions in ASSER, and whether a version is valid to return is determined by the consistency level that ASSER is configured as. We implement two consistency levels in the current ASSER: sequential consistency and PRAM (Pipeline RAM) consistency. These two consistency levels represent reverse consistency-performance tradeoff, and ASSER naturally supports both of them to meet different requirement in varied applications or services. Based on MPL mechanism, we respectively design dedicated read/write protocols for sequential and PRAM consistency.

Object-based Storage
PRAM consistency is identified as the minimum consistency level that maintains reasonable semantics without resulting in excessive complexity for system developers. PRAM consistency only ensures that, all write operations originated by a single client are seen by all other clients in the order in which they were performed, as if all the write operations performed by a single client are in a pipeline. Write operations issued by different clients may be seen by different clients in different orders. Moreover, PRAM consistency implemented within ASSER guarantees read-yourwrites semantics, which means the effect of a write operation by a client on object f will always be seen by a successive read operation on f by the same client

IV. FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.
This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system. The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

V. CONCLUSION

This paper presented the design and implementation of disease a hybrid storage scheme that aims at providing balanced trade-off between I/O performance and space efficiency with low storage cost. We proposed a mechanism called multiversional parity logging to facilitate efficient read/write handling in disease. Furthermore, we also described how sequential and PRAM consistency are provided based on the elementary read/write protocols. We evaluated the performance of disease and the robustness of its implementation. According to our experimental results, with only half of extra space overhead, disease outperformed CRAQ in write-heavier workload and stayed evenly matched in read-heavier workload.

Disease also surpassed pure Reed-Solomon code in terms of read performance, and remained competitive with Rscode when handling write requests. Besides, we conducted sensitivity experiments to demonstrate the stability and consistency of disease's performance over varied workloads and parameter settings. Finally, we verified the feasibility of disease in practical environment through real-world traces driven experiment. The newly developed system, in its present form, is eminently suited to the existing needs. But in order to meet the future needs, which can become progressively more complex the efficiency of the system can be improved by making some simple modifications in the programs.

The project is still developing in many ways. The site is being added new section and new services to the visitors. In future the site will be providing the service of modifying the website to the users. The user is given a blank website so as to design the website or web page of his own, by activating links, changing images and changing the text using editor. In future the project will be altered by which the user is given more options to design more number of pages. This will be achieved by providing more links ,more customizing options and we may offer the user possibility of creating domain of its won.

VI. ACKNOWLEDGEMENT

This research was supported by my guide and supervisors. I thank my faculties and friends who provided insight and expertise that greatly assisted the research, although they may not agree with all of the interpretations of this paper. I thank Mrs. Chandra Prabha. K, Assistant Professor, Alagappa Chettiar Government College of Engineering & Technology for commenting that greatly improved the manuscript.

REFERENCES

[1] Ceph, "Rados in ceph , http://ceph.com/ceph-storage/block-storage/."

[2] Amazon, "Amazon simple storage service faqs, http://aws.amazon.com/s3/faqs/."

[3] S. Ghemawat, H. Gobioff, and S. Leung, "The google file system," in Proceedings of the 9th ACM SOSP, 2003.

[4] H. C. Chen, Y. Hu, P. P. Lee, and Y. Tang, "Nccloud: A networkcoding- based storage system in a cloud-of-clouds," IEEE Transactions on Computers, vol. 63, no. 1, 2014.

[5] R. Li, J. Lin, and P. P. C. Lee, "Enabling concurrent failure recovery for regenerating-coding-based storage systems: From theory to practice," IEEE Transactions on Computers, vol. 64, no. 7, 2015.

[6] R. Vinayak, Erasure Coding for Big-data Systems: Theory and Practice. Technical Report No. UCB/EECS-2016-155, University of California, Berkeley, 2016.

[7] R. V. Renesse and F. B. Schneider, "Chain replication for supporting high throughput and availability," in Proceedings of 6th OSDI, 2004.

[8] HDFS-RAID, "Distributed raid file system, http://wiki.apache.org/hadoop/hdfs-raid."

[9] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows azure storage," in Proceedings of USENIX Annual Technical Conference. USENIX, 2012, pp. 15–26.

[10] A. Fikes, "Storage architecture and challenges," in Google Faculty Summit, 2010.

[11] Y. Zhu, J. Lin, P. P. C. Lee, and Y. Xu, "Boosting degraded reads in heterogeneous erasure-coded storage systems," IEEE Transactions on Computers, vol. 64, no. 8, 2015.

[12] J. Huang, X. Qin, X. Liang, and C. Xie, "An efficient i/o-redirection based reconstruction scheme for erasure-coded storage clusters," IEEE Transactions on Computers, vol. 64, no. 11, 2015.