International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 6 Issue: 6 254 - 259

PAS: A Sampling Based Similarity Identification Algorithm for compression of Unicode data content

Siddiqui Sara Begum Anjum Parvez
Department of Computer Science and Engineering, K.S.I.E.T Hingoli-431513 Maharashtra, INDIA.
E-mail id: siddiqui123sara@gmail.com

Ashru L. Korde *(Author)*
Department of Computer Science and Engineering, K.S.I.E.T Hingoli-431513 Maharashtra, INDIA.
Email id:ashrukorde@gmail.com

***Abstract*—** Generally, Users perform searches to satisfy their information needs. Now a day's lots of people are using search engine to satisfy information need. Server search is one of the techniques of searching the information. the Growth of data brings new changes in Server. The data usually proposed in timely fashion in server. If there is increase in latency then it may cause a massive loss to the enterprises. The similarity detection plays very important role in data. while there are many algorithms are used for similarity detection such as Shingle, Simhas TSA and Position Aware sampling algorithm. The Shingle Simhash and Traits read entire files to calculate similar values. It requires the long delay in growth of data set value. instead of reading entire Files PAS sample some data in the form of Unicode to calculate similarity characteristic value.PAS is the advance technique of TSA. However slight modification of file will trigger the position of file content .Therefore the failure of similarity identification is there due to some modifications.. This paper proposes an Enhanced Position-Aware Sampling algorithm (EPAS) to identify file similarity for the Server. EPAS concurrently samples data blocks from the modulated file to avoid the position shift by the modifications. While there is an metric is proposed to measure the similarity between different files and make the possible detection probability close to the actual probability. In this paper describes a PAS algorithm to reduce the time overhead of similarity detection. Using PAS algorithm we can reduce the complication and time for identifying the similarity. Our result demonstrate that the EPAS significantly outperforms the existing well known algorithms in terms of time. Therefore, it is an effective approach of similarity identification for the Server.

Keywords*: Similarity, identification, Unicode data*.

*****

I. INTRODUCTION

The growth of the data management significantly increases and the data risk and cost of data also increases . to address this kind of problem many users transfer there data to the server. And we can access that data via internet. This kind of problem result in large volume of redundant data in server. The main reson for this is that multiple users tends to store similar files in the server . Here multiple users store multiple files in the server. Unfortunately the redundant data not only consume significant it esurses but also occupy bandwidth for this puposes the data deduplication is required .

To overcome from all these problems we create the sampling similarity based identification algorithm for compression of Unicode data content in server. search techniques perform comparably despite contrary claims in the literature. During my evaluation of search effectiveness, I were surprised by the difficulty I had searching my data sets. In particular, straightforward implementations of many search techniques server not scale to databases with hundreds of thousands of tuples, which forced us to write "lazy" versions of their core algorithms and reduce their memory footprint. Even then, I were surprised by the excessive runtime of many search techniques.

In the present data warehousing environment schemes there are lots of issues in server computing. Some advanced data manipulating schemes are required to extending the dats search paradigm to relational data has been an active area of research within the database and information retrieval community. It proves that the effectiveness of performance in retrieval tasks and data maintaining procedures. The outcome confirms previous claims regarding the unacceptable performance of these systems and underscores the need for standardization as exemplified by the IR community when evaluating these retrieval systems.

Position Aware similarity identification algorithms belong to I/O bound and CPU bound tasks. Calculating the Unicode of similar files requires lots of CPU Corresponding cycles, the computing increases with the growth of data sets

Position Aware similarity identify algorithms normally require a large amount of time for detecting the similarity, which results in long delays and if there is large data sets. it require more time This makes it difficult to apply the algorithms to some applications.

In this paper, we propose a Position-Aware Similarity (PAS) identification algorithm to detect the similar files in large data sets. This method is very effective in dealing with file modification when performing similarity detection. And here we use Simhash algorithm called in terms of precision and recall. Furthermore, the time overhead, CPU and memory occupation of PAS are much less than that of simhash. This is because the overhead of PAS is relatively stable. It is not increases with the growth of data size.

The remainder of this paper is organized as follows: we present related work in section 2. In section 3 we describe some background knowledge. Section 4 introduces the basic idea of PAS algorithm.Section 5 shows Sampling Based similarity identification. Section 6 shows the evaluation results of PAS algorithm. Section 7 shows Similarity Identification Techniques Work Section 8 draws conclusions and Future use.

II. RELATED WORK

In related work it involved the Server technique and similarity detection algorithm to avoid the redundant data in server using Unicode data content. Here we design the sampling based similarity approach for the detection of data similarity and perform the various task like upload file and delete files

In the past decade, a lot of research efforts have been invested in identifying data similarity.Which we explain below.

The first one is similar web page detection with web search engine. Detecting and removing similar web pages can save network bandwidth, reduce storage consumption, and improve the quality of web search engine index. Andrei et al. [17], [20] proposed a similar web page detection technique called Shingle algorithm which utilizes set operation to detect similarity. Shingle is a typical sampling based approach employed to identify similar web pages. In order to reduce the size of shingle, Andrei presented Modm and Mins sampling methods. This algorithm is applied to AltaVista web search engine at present. Manku et al. [21] applied a Simhash algorithm to detect similarity in web documents belonging to a multi-billion page repository. Simhash algorithm practically runs at Google web search engine combining with Google file system [22] and MapReduce [23] to achieve batch queries. Elsayed et al. [24] presented a MapReduce algorithm for computing pairwise document similarity in large document collections.

The second one is similar file detection in storage systems. In storage systems, data similarity detection and encoding play a crucial role in improving the resource utilization. Forman [25] presented an approach for finding similar files and applied the method to document repositories.This approach brings a great reduction in storage space consumption. Ouyang [26] presented a large-scale file compression technique based on cluster by using Shingle similarity detection technique. Ouyang uses Min-wise [27] sampling method to reduce the overhead of Shingle algorithm. Han et al. [28] presented fuzzy file block matching technique, which was first proposed for opportunistic use of content addressable storage. Fuzzy file block matching technique employs Shingle to represent the fuzzy hashing of file blocks for similarity detection. It uses Mins sampling method to decrease the overhead of shingling algorithm.

The third one is plagiarism detection. Digital information can be easily copied and retransmitted. This feature causes owners copyright be easily violated. In purpose of protecting copyright and other related rights, we need plagiarism detection. Baker [29] described a program called dup which can be used to locate instances of duplication or near duplication in a software. Shivakumar [30] presented data structures for finding overlap between documents and implemented these data structures in SCAM.

The forth one is remote file backup. Traditional remote file backup approaches take high bandwidth and consume a lot of resources. Applying similarity detection to remote file backup can greatly reduce bandwidth consumption.

Teodosiu et al. [15] proposed a Traits algorithm to find out the client files which are similar to a given server file. Teodosiu implemented this algorithm in DFSR. Experimental results suggest that these optimizations may help reduce the bandwidth required to transfer file updates across a network. Muthitacharoen et al. [5] presented LBFS which exploits similarity between files or versions of the same file to save bandwidth. Cox et al. [31] presented a similaritybased mechanism for locating a single source file to perform peer-to-peer backup. They implemented a system prototype called Pastiche.

The fifth one is the similarity detection for specific domains. Hua et al. [32] explored and exploited data similarity which supports efficient data placement for cloud. They designed a novel multi-core-enabled and localitysensitive hashing that can accurately capture the differentiated

similarity across data. Biswas et al. [33] proposed a cache architecture called Mergeable. Mergeable detectsdata similarities and merges cache blocks so as to decrease cache storage requirements. Experimental evaluation suggested that Mergeable reduces off-chip memory accesses and overall power usage. Vernica et al. [34] proposed a three-stage approach for end-to-end setsimilarity joins in parallel using the popular MapReduce framework. Deng et al. [35] proposed a MapReducebased framework Massjoin for scalable string similarity joins. The approach achieves both set-based similarity functions and character-based similarity

functions. Most of the above work focus on a specific application scenario, and the computational or similarity detection overhead are increased with the growth of data volume. In addition, the similarity detection metric may not be able to well measure the similarity between two files. Therefore, this paper proposes an EPAS algorithm and a new similarity detection metric to identify file similarity for the cloud.

According to the analysis (see Section 4 and 5.3) and experimental results, it illustrates that the proposed similarity metric catch the similarity between metric catch the similarity between two files more accuratelythat that of traditional metric. Furthermore, the overhead of EPAS is fixed and minimized in contrast to previous work.

III. BACKGROUND

Here We used the hash key which is based on counting the occurrences of certain Unicode strings within a file. The keys, along with certain intermediate data, are stored in a relational database (Figure 1).A separate program then query the database for keys with similar values, and outputs the results. Our code was written in PHP, and developed simultaneously for the Windows platforms. While the code runs equally well on platforms, we used a Windows machine for primary development and for most data collection.

File Unicode data

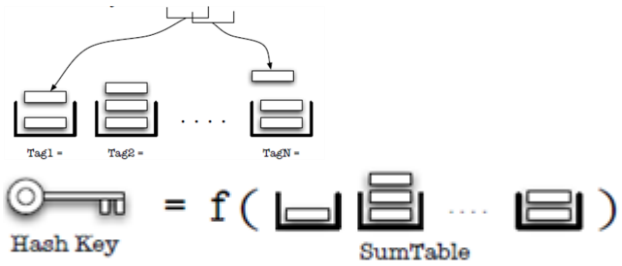

Figure 1: SimHash produces two levels of file similarity data.
the key function was implemented to account for file extensions.

Here We can not say that two files are different if they contain different extensions. Assign different values to any two files with different extension. To this we compute a hash to file extension with value between 0 and 1.if there is same extension then this will not affect the relation between files with same extension .

I. POSITION-AWARE SIMILARITY ALGORITH

In server if there is large similar data set with less overhead. Then we can use the similarity detection algorithm that is PAS. Here we use different symbols which are given in the table bellow.

A. Traditional sampling algorithm

Suppose we sample N data blocks of file A, each data block sizing Lenc is injected to a hash function. We then can obtain N fingerprint values that are collected as a fingerprint set SigA(N; Lenc). In this scenario, similarity detection problem can be transformed into a set intersection problem. By analogy, we will have a fingerprint set SigB(N; Lenc) of file According to equation (1),

Symbol	Meaning
Lenc	The length of sampling data blocks length
N	The number of sampling data blocks
FileSize	File size
LenR	The distance between two sampling data blocks
T	Sampling position impact factor of PAS
δ	The threshold of PAS

Table 1: Symbols used in following

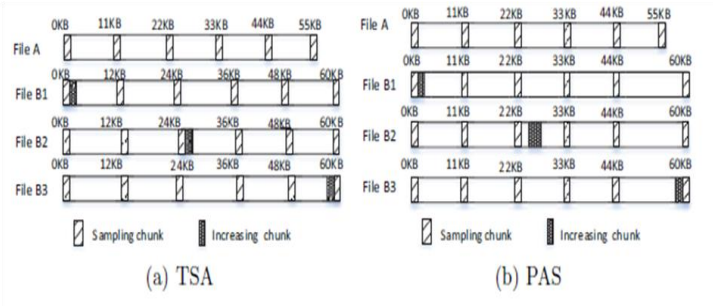

FIGURE. 2: THE SAMPLING POSITIONS OF TSA AND PAS

the degree of similarity between file A and file B can be described as equation (1), where Sim(A;B) ranges s between 0 and 1. If Sim(A;B) is reaching 1, it means the similarity of file A and file B are very high, vice verse. After selecting a threshold of the similarity, we can determine that file A is similar to file B when Sim(A;B) is satisfied. This TSA is described in algorithm 1 by using pseudo-code.

$$Sim(A,B) = \frac{|Sig_A(N,Lenc) \cap Sig_B(N,Lenc)|}{|Sig_A(N,Lenc) \cup Sig_B(N,Lenc)|}$$

TSA is simple, but it is very sensitive to file modifications. A small modification would cause the sampling positions shifted, thus resulting a failure. Suppose we have a file A sizing 56KB. We sample 6 data blocks and each data block sizes 1KB. According to Algorithm 1, file A has N= 6; Lenc = 1KB; FileSize = 56KB; LenR = 10KB. If we add 5KB data to file A to form file B, file Bwill have N = 6; Lenc = 1KB; FileSize = 61KB; LenR = 11KB in terms of algorithm.

Algorithm 1 Traditional Sampling Algorithm:

```
function TRADITIONALSAMPLING(fd, N, Lenc)
    LenR = (FileSize - Lenc*N)/(N - 1)//Calculate distance between the sampling
data blocks
    for i = 1 to N do
        offset = (i - 1)*(Lenc + LenR)//Calculate the sampling offset
        lseek(fd, offset, SEEK_SET) //Set the sampling offset
        read(fd, buf, Lenc)
        Md5(buf, Lenc, Md5Val)
        put(Md5Val, SigA)          //Put the fingerprint to the Sig_A(N, Lenc)
    end for
end function
```

Adding 5KB data to file A has three situations including the begging, the middle, and the end of the file A. File B1, B2, and B3 in figure 2(a) represent

Algorithm 2 PAS Sampling Algorithm:

```
function PASSAMPLING(fd, N, Lenc, T)
    FileSize = (FileSize/T)T
    LenR = (FileSize - Lenc*N)/(N - 1)
    LenR = LenR > 0 ? LenR : 0
    for i = 1 to N - 1 do
        offset = (i - 1)*(Lenc + LenR)
        lseek(fd, offset, SEEK_SET)
        read(fd, buf, Lenc)
        Md5(buf, Lenc, Md5Val)
        put(Md5Val, SigA)
    end for
    lseek(fd, -Lenc, SEEK_END)
    read(fd, buf, lenc)
    Md5(buf, Lenc, Md5Val)
    put(Md5Val, SigA)
end function
```

these three different situations. We can find that the above file modifications cause the sampling position shifted and result in an inaccuracy of similarity detection.

For example, the six sampling positions of file A are 0KB, 11KB, 22KB, 33KB, 44KB, and 55KB ((1 □ 1) _ (1 + 10) = 0KB; (2□ 1) _ (1 +10) = 11KB; (3 □ 1) _ (1 + 10) = 22KB; (4 □ 1) (1 +10) = 33KB; (5 □1) _ (1 + 10) = 44KB; (6 □ 1) _ (1 + 10) = 55KB), respectively.

However, due to the added 5KB data, the six sampling positions of file B1,B2, and B3 are shifted to 0KB; 12KB; 24KB; 36KB; 48KB, and 60KB((1 □ 1) _ (1 + 11) =0KB; (2 □ 1) _ (1 + 11) = 12KB; (3 □1) _ (1 + 11) = 24KB; (4 □ 1) _ (1 + 11) =36KB; (5 □ 1) _ (1 + 11) = 48KB; (6 □ 1) _ (1 + 11) = 60KB), respectively.

Although the Sim(A;B) is far from actual value when using TSA, the sampling method is very simple and takes

much less overhead in contrast to the shingle algorithm and simhash algorithm.

B. PAS algorithm

FPP[17] exploits prefetching fingerprints belonging to the same file by leveraging file similarity, thus improving the performance of data deduplication systems.

The experimental results suggest that FPP increases cache hit ratio and reduces the number of disk accesses greatly. FPP samples three data blocks in the beginning, the middle, and the end of files to determine that a forthcoming file is similar to the files stored in the backed storage system, by using the TSA. This method is sample and effective. However, as explained in section 4.1, a single bit modification would result in a failure. Therefore, PAS is proposed to solve this problem.

II. CONCLUSION AND FUTURE SCOPE

A. Conclusion

In this paper , Overall we will study all the existing techniques which is available in market. Each system has some advantages and some disadvantages. Any existing system cannot fulfill all the requirement of Server search. They require more space and time; also some techniques are limited for particular dataset. We Proposed an algorithm PAS to identify the file similarity of Unicode data in large data Set.Here many experiments are performed to select the parameters of PAS. PAS is very effective in detecting file similarity in contrast in similarity identification algorithm called Simhas. PAS required less time than Simhash. The Proposed technique is satisfying number of requirement of server search using different algorithms. It also shows the ranking of character value and not requires the knowledge of database queries. Compare to existing algorithm it is a fast process.

B. Future Scope

As a future work we can search the techniques which are useful for all the datasets, means only single technique can be use. Further research is necessary to investigate the experimental design decisions that have a significant impact on the evaluation of server search.

Sampling based similarity identification opens up several directions for future work. The first one is using content based chunk algorithm to sample data blocks, since this approach can avoid content shifting incurred by data modification. The second one is employing file metadata to optimize the similarity detection. This is because the file size and type which are contained in the metadata of similar file are normally very close.

Evaluate to presented systems it is a fast process and the Techniques are implausible to have performance characteristics that are similar to existing systems but be required to be used if cloud search systems are to scale to great datasets. The memory exploitation during a search has not been the focus of any earlier assessment

ACKNOWLEDGMENT

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression, “One of us (R.B.G.) thanks . . .” Instead, try

"R.B.G. thanks". Put applicable sponsor acknowledgments here; DO NOT place them on the first page of your paper or as a footnote.

REFERENCES

[1] 1.storage architecture for big data in cloud environment," in Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing. IEEE, 2013, pp. 476– 480.

[2] J. Gantz and D. Reinsel, "The digital universe decade-are you ready," IDC iView, 2010.

[3] H. Biggar, "Experiencing data de-duplication: Improving efficiency and reducing capacity requirements," The Enterprise Strategy Group, 2007.

[4] F. Guo and P. Efstathopoulos, "Building a high performance deduplication system," in Proceedings of the 2011 USENIX conference on USENIX annual technical conference. USENIX Association, 2011,pp. 25–25.

[5] A. Muthitacharoen, B. Chen, and D. Mazieres, "A low-bandwidth network file system," in ACM SIGOPS Operating Systems Review, vol. 35, no. 5. ACM, 2001, pp. 174–187.

[6] B. Zhu, K. Li, and R. H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system." in Fast, vol. 8, 2008, pp. 1–14.

[7] Y. Deng, "What is the future of disk drives, death or rebirth?" ACM Computing Surveys (CSUR), vol. 43, no. 3, p. 23, 2011.

[8] C. Wu, X. LIN, D. Yu, W. Xu, and L. Li, "End-to-end delay minimization for scientific workflows in clouds under budget constraint," IEEE Transaction on Cloud Computing (TCC), vol. 3, pp. 169–181, 2014.

[9] G. Linden, "Make data useful," http://home.blarg.net/_glinden/ StanfordDataMining.2006-11-29.ppt, 2006.

[10] R. Kohavi, R. M. Henne, and D. Sommerfield, "Practical guide to controlled experiments on the web: listen to your customers not to the hippo," in Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007, pp. 959–967.

[11] J. Hamilton, "The cost of latency," Perspectives Blog, 2009.

[12] D. Bhagwat, K. Eshghi, D. D. Long, and M. Lillibridge, "Extreme binning: Scalable, parallel deduplication for chunk-based file backup," in Modeling, Analysis & Simulation of Computer and Telecommunication Systems, 2009. MASCOTS'09. IEEE International Symposium on. IEEE, 2009, pp.

[13] W. Xia, H. Jiang, D. Feng, and Y. Hua, "Silo: a similarity-locality based near-exact deduplication scheme with low ram overhead and high throughput," in Proceedings of the 2011 USENIX conference on USENIX annual technical conference. USENIX Association, 2011, pp. 26–28.

[14] Y. Fu, H. Jiang, and N. Xiao, "A scalable inline cluster deduplication framework for big data protection," in Middleware 2012.Springer, 2012, pp. 354–373.

[15] D. Teodosiu, N. Bjorner, Y. Gurevich, M. Manasse, and J. Porkka, "Optimizing file replication over limited bandwidth networks using remote differential compression," Microsoft Research TR- 2006-157, 2006.

[16] Y. Zhou, Y. Deng, and J. Xie, "Leverage similarity and locality to enhance fingerprint perfecting of data deduplication," in Proceedings of The 20th IEEE International Conference on Parallel and Distributed Systems. Springer, 2014.

[17] A. Z. Broder, "On the resemblance and containment of documents," in Compression and Complexity of Sequences 1997. Proceedings. IEEE, 1997, pp. 21–29.

[18] G. S. Manku, A. Jain, and A. Das Sarma, "Detecting near duplicates for web crawling," in Proceedings of the 16th international conference on World Wide Web. ACM, 2007, pp. 141–150.

[19] L. Song, Y. Deng, and J. Xie, "Exploiting fingerprint prefetching to improve the performance of data deduplication," in Proceedings of the 15th IEEE International Conference on High Performance Computing and Communications. IEEE, 2013.

[20] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, "Syntactic clustering of the web," Computer Networks and ISDN Systems, vol. 29, no. 8, pp. 1157–1166, 1997.

[21] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. ACM, 2002, pp. 380–388.

[22] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in ACM SIGOPS Operating Systems Review, vol. 37, no. 5. ACM, 2003, pp. 29–43.

[23] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.

[24] T. Elsayed, J. Lin, and D. W. Oard, "Pairwise document similarity in large collections with map reduce," in Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers. Association for Computational Linguistics, 2008, pp. 265–268.

[25] G. Forman, K. Eshghi, and S. Chiocchetti, "Finding similar files in large document repositories," in Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. ACM, 2005, pp. 394–400.

[26] Z. Ouyang, N. Memon, T. Suel, and D. Trendafilov, "Cluster-based delta compression of a collection of files," in Web Information Systems Engineering, 2002. WISE 2002. Proceedings of the Third International Conference on. IEEE, 2002, pp. 257–266.

[27] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," Journal of Computer and System Sciences, vol. 60, no. 3, pp. 630–659, 2000.

[28] B. Han and P. J. Keleher, "Implementation and performance evaluation of fuzzy file block matching." in USENIX Annual Technical Conference, 2007, pp. 199–204.

[29] B. S. Baker, "On finding duplication and near-duplication in large software systems," in Reverse Engineering, 1995., Proceedings of 2nd Working Conference on. IEEE, 1995, pp. 86–95.

[30] N. Shivakumar and H. Garcia-Molina, "Building a scalable and accurate copy detection mechanism," in Proceedings of the first ACM international conference on Digital libraries. ACM, 1996, pp.160–168.