International Journal of Computer Science Trends and Technology (IJCST) – Volume 6 Issue 2, Mar - Apr 2018

RESEARCH ARTICLE OPEN ACCESS

Achieving Space-Efficiency And Access Isolation By Implementing File Partition For Online Data

Ankita Bung [1], C.Sudha [2], Sravanthi Nanumasa [3]
Assistant professor
Dept of CSE,Mahatma Gandhi Institute of Technology
Hyderabad – India

ABSTRACT
Given a fixed of files that show a certain degree of similarity, we take into account a novel problem of performing statistics redundancy elimination throughout a hard and fast of distributed nodes in a shared-not anything in-reminiscence big data analytic system. The redundancy elimination scheme is designed in the following methods that is first space utilization: the total area needed to keep the files is minimized and second access-isolation: data shuffling amongst server is likewise minimized. In this paper work , we first analyze that locating an access-efficient and area most useful solution is an enhanced NP-Hard problem. Following this, we present the Advanced file partitioning Algorithm that locate access utilization solutions in an incremental way with minimal set of rules time complexity ie polynomial time. Our experimental techniques on more than one records units confirms that the proposed record partitioning solution is able to acquire compression ratio close to the greatest compression overall performance achieved with the aid of a centralized solution.

Keywords :– *Big data,NP-Hard,File Partitioning Algorithm*

I. INTRODUCTION

Big Data analytics is an umbrella term, that includes tactics and technology, hardware and application program for collecting, coping with and analyzing giant scale based and unstructured data in real world data. Tremendous facts analytics works on whole records as opposed to handiest pattern knowledge in traditional expertise analytics schemes. Within the case of small documents, analyses had been accomplished by way of randomly making a choice on samples (partial data) which have been regarded as consultant of the entire documents. Because of evaluation of nice partial information, the know-how extracted are erroneous and incomplete and hence the choices made are sub-most reliable and the total performance accomplished are terrible and sub top-quality. Particularly in the case of actual community analysis and troubleshooting, special and brief info are desired for providing actual solution, which would nice be viable if whole/significant documents is analyzed.In these days we're witnessing Jim gray's perception that "memory is the new disk[1] and
disk is the brand new tape becoming real. Extra and bigger scale functions now depend on low cost reminiscence (including random access memory RAM and nonvolatile reminiscence akin to stable-state disk) in pc cluster to cache, trade, retailer, and procedure information in both significant and disbursed method. Chiefly within the massive-information era, in-reminiscence information processing/analytic methods comparable to Spark, Storm, Map curb online, Picoolo, SINGA, Pregel and GraphLab, are becoming increasingly widespread as consumers are annoying rapid and reactive data analytics. With an remarkable expense at which the data volumes are increasing, how do these in-memory methods efficiently utilize the restricted memory in a allotted computer cluster becomes an primary study situation. In this paper, we primarily lift two design issues for the shared-nothing disbursed in-reminiscence programs as follows. (a) what is the mostspace-effective technique to distribute information amongst servers? (b) how one can limit knowledge shuffling among servers? For the reason that the traits of a memory method (e.G., high I/O bandwidth, good reliability, moderately confined size), a space-effective knowledge compression scheme that minimizes data redundancy permits the server to retailer as much know-how as possible within the memory of each and every server.

space effiency is most often executed by way of the method of De-duplication, which splits all of the records in the

system into chunks and maintains best a unique copy of every chunk.De-duplication operates most often achieves 30% - 50% compression in quite a lot of application data. Nonetheless, most knowledge De-duplication options have fascinated about reducing the distance-overhead within a single server. These methods do not take into account the challenge of information De-duplication in a allotted atmosphere on account that of the high network access cost and knowledge concurrency manipulate overhead in a disbursed resolution.Without careful similarity evaluation for the set of documents that must be processed via more than one servers, precise data chunk may be stored in exceptional servers. Recreating thelong-established knowledge could require a big quantity of data shuffling operations among servers,which can significantly slow down the procedure. An access-efficient answer is competent of minimizing information shuffling via storing the specified chunks which can be required to recreate a file in the equal server. Given a set of files that show a specified degree of similarity or redundancy, we recall the main issue of assigning them to a collection of allotted servers
and gift options which can be both access isolated and space-efficient.

II. LITERATURE SURVEY

A Study of Practical De-duplication

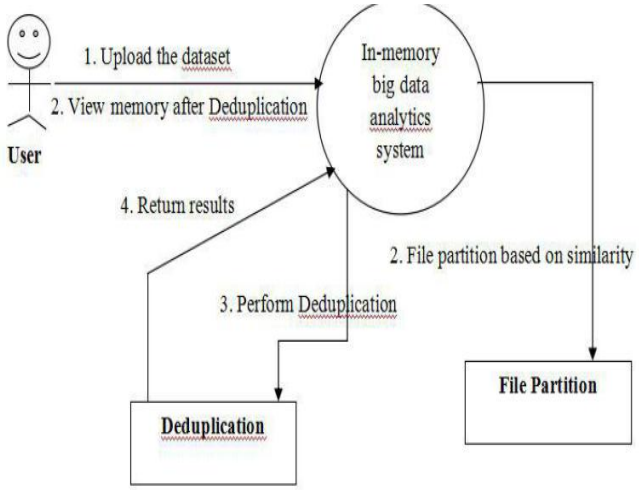
We collected file system content data from 857 desktop computers at Microsoft over a span of 4 weeks. We analyzed the data to determine the relative efficacy of data de-duplication, particularly considering whole-file versus block-level elimination of redundancy. We found that whole-file de-duplication achieves about three quarters of the space savings of the most aggressive block-level de-duplication for storage of live file systems, and 87% of the savings for backup images[10]. We also studied file fragmentation finding that it is not prevalent, and updated prior file system metadata studies, finding that the distribution of file sizes continues to skew toward very large unstructured files. We studied file system data, metadata, and layout on nearly one thousand Windows file systems in a commercial environment. This new dataset contains metadata records of interest to file system designers; data content findings that will help create space efficiency techniques and data layout information useful in the evaluation and optimization of storage systems. We find that whole-file de-duplication together with sparseness is a highly efficient means of lowering storage consumption, even in a backup scenario. It approaches the effectiveness of conventional de-duplication at a much lower cost in performance and complexity. The environment we studied, despite being homogeneous, shows a large diversity in file system and file sizes.These challenges, the increase in unstructured files and an ever-deepening and more populated namespace pose significant challenge for future file system designs. However, at least one problem – that of file fragmentation, appears to be solved, provided that a machine has periods of inactivity in which defragmentation can be run.

Decentralized De-duplication in SAN Cluster File Systems

File systems hosting virtual machines typically contain many duplicated blocks of data resulting in wasted storage space and increased storage array cache footprint.De-duplication addresses these problems by storing a single instance of each unique data block and sharing it between all original sources of that data. While De-duplication is well understood for file systems with a centralized component, we investigate it in a decentralized cluster file system, specifically in the context of VM storage. We propose DEDE, a block-level De-duplication system for live cluster file systems that does not require any central coordination, tolerates host failures, and takes advantage of the block layout policies of an existing cluster file system. In DEDE, hosts keep summaries of their own writes to the cluster file system in shared on-disk logs[11]. Each host periodically and independently processes the summaries of its locked files, merges them with a shared index of blocks, and reclaims any duplicate blocks. DEDE manipulates metadata using general file system interfaces without knowledge of the file system implementation. We present the design, implementation, and evaluation of our techniques in the context of VMware ESX Server. Our results show an 80% reduction in space with minor performance overhead for realistic workloads. In this paper, we studied De-duplication in the context of decentralized cluster file systems. We have described a novel software system, DEDE, which provides block level De-duplication of a live, shared file system without any central coordination. Furthermore, DEDE builds atop an existing file system without violating the file system's abstractions, allowing it to take advantage of regular

file system block layout policies and in-place updates to unique data. Using our prototype implementation, we demonstrated that this approach can achieve up to 80% space reduction with minor performance overhead on realistic workloads. We believe our techniques are applicable beyond virtual machine storage and plan to examine DEDE in other settings in the future. We also plan to explore alternate indexing schemes that allow for greater control of De-duplication policy. For example, high-frequency De-duplication could prevent temporary file system bloat during operations that produce large amounts of duplicate data (e.g., mass software updates), and deferral of merge operations could help reduce file system fragmentation.

III. ARCHITECTURE



The system architecture where all the objects areconnected to the core which is the Big data analytics system and the operations performed by the user are file partitioning and deduplication

IV. SYSTEM WORK

Initial File Partition

In the initial file partition it groups similar files together can effectively reduce the amount of unique data chunks that have to be sent to the underneath distributed storage and to be loaded into the memory.

Enhanced Initial file partitioning Algorithm:

Algorithm for Similarity-aware-partitioning Algorithm
1: Input: Set of files f = {f1,f2, . . . fn} and capacity C
2: Construct G(f)
3: Construct T upon G(f)
4: for every node i do
5: for every subtree τj rooted at the neighbor j do
6: Execute Flood(τj, P(Δ(Ai)) − wi,j)
7: end for
8: end for
9: while size of T is larger than C do
10: Select a new worker node
11: Execute TreeP art(T,C)
12: end while
13: Assign remaining T to a worker node
14: for each worker node do
15: Perform data compression independently
16: end for
Flooding Subroutine Flood(τi,Δγ)
17: Input: Tree τi and message value Δγ
18: Node i sends Δγ to all of its neighbors in τi
19: for each node j that receives Δγ from node k do
20: Update γj(k) = γj(k) + Δγ
21: Forward the message to neighbor nodes in τi except node k
22: end for
Tree Partitioning Subroutine TreeP art(N, CR)
23: Input: Node set N and residual capacity CR
24: if Successfully locate a node i in N that fulfills (1) then
25: Assign components of τi including the largest component to worker nodes greedily
26: Prune off the assigned components
27: Reattach the disconnected components to T
28: Update γ using the Flooding Subroutine
29: Update CR = CR − (size of the pruned components)
30: Recursively execute TreeP art(N, CR) on the unassigned components of τi
31: end if

Incremental File Partition

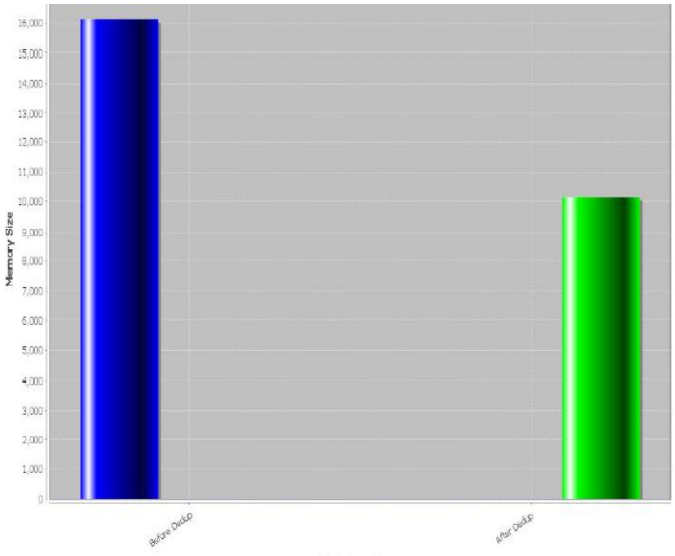
An incremental file partition that further removes redundant data from the newly

added data can potentially save the precious memory space

Incremental File partition Algorithm:

1: Input new Set of files f = *{f*1,*f*2, . . . *fn}*
2: set value of *j* to null
3: for every node *i*
4: get the chunk size in integer data type
5: read the value of chunk size
6: obtain the file pointer
7: set flag value as false
8: for every *k*
9: find the string similarity value
10: obtain the value of *compare*
11: if *compare* > 0.90 set flag to true
12: if not, then
13: get path and the length
14:close and increment *j* by 1

V. EXPERIMENTAL RESULTS



The above fig shows the graph before and after deduplication along with the units on the Y-axis which constitutes the memory size of the data set.

VII. CONCLUSION AND FUTURE WORK

In this paper, we consider the problem of storing a set of files A that exhibit certain degree of similarity among distributed worker nodes in a shared-nothing in-memory big-data analytic system. We successfully propose Enhanced file partitioning algorithm that achieves both space-efficiency and access-isolation.Theoretical performance bounds of the proposed algorithm is presented and the experimental results confirm the insights we obtained. In the future we wish to consider the morel cases of fault tolerance.load-balance and multiple-server communication to explore new trade-offs.

REFERRENCES

[1] Online Data Deduplication for In-Memory Big-data analytic systems,IEEE 2017.

[2] S. Robbins, “Ram is the new disk,” in InfoQ News, 2008.

[3] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J.Franklin,S. Shenker, and I. Stoica, “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing,” in Presented as part of the 9th USENIXSymposium on Networked Systems Design and Implementation (NSDI 12), pp.15–28, USENIX, 2012.

[4] BackType and Twitter,“Storm:Distributed and fault-tolerant realtime computation.” [Online] Available: http://storm.apache.org/.

[5] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears,“Mapreduce online,” in Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation,NSDI’10,(Berkeley, CA, USA), pp. 21–21,USENIX Association, 2010.

[6] R. Power and J. Li, “Piccolo: Building fast, distributed programs with partitioned tables,” in Proceedings of the 9th USENIX Conference on

Operating Systems Design and Implementation, OSDI'10, (Berkeley,CA, USA), pp. 1–14, USENIX Association, 2010.

[7] S. TEAM, "Singa: A distributed training platform for deep learning models."[Online] Available: http://www.comp.nus.edu.sg/~dbsystem/singa/.

[8] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser,and G. Czajkowski, "Pregel: A system for large-scale graph processing,"in Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD '10, (New York, NY, USA), pp. 135–146, ACM, 2010.

[9] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M.Hellerstein, "Distributed graphlab: A framework for machine learning and data mining in the cloud," Proc. VLDB Endow., vol. 5, pp. 716–727,Apr. 2012.

[10] D. T. Meyer and W. J. Bolosky, "A study of practical deduplication,"Trans.Storage, vol. 7, pp. 14:1–14:20, Feb. 2012.

[11] A. T. Clements, I. Ahmad, M. Vilayannur, and J. Li, "Decentralized deduplication in san cluster file systems," in Proceedings of the 2009 conference on USENIX Annual technical conference, USENIX'09,(Berkeley, CA, USA), pp. 8–8, USENIX Association, 2009.

[12] U. Manber, "Finding similar files in a large file system," in Proceedings of the USENIX Winter 1994 Technical Conference on USENIX Winter1994 Technical Conference, WTEC'94, (Berkeley, CA, USA), pp. 2–2,USENIX Association, 1994. 35

[13] B. S. Baker, "On finding duplication and near-duplication in large software systems," in Proceedings of the Second Working Conference on Reverse Engineering, WCRE '95, (Washington, DC, USA), pp. 86–,IEEE Computer Society, 1995.

[14] 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, KDD '05, (New York, NY, USA), pp. 394–400, ACM, 2005.

[15] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer,"Reclaiming Space from Duplicate Files in a Serverless Distributed File System," in Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS '02), (Vienna, Austria),pp. 617–624, July 2002.

[16] S. Quinlan and S. Dorward, "Venti: A new approach to archival data storage," in Proceedings of the 1st USENIX Conference on File and Storage Technologies, FAST '02, (Berkeley, CA, USA), USENIX Association, 2002.load-balance and multiple-server communication to explore new trade offs.